Concluding Exercise in R

Sharon Hadar

09.13.2017

Execrise #1:

In this execrise you will perform a Monte Carlo study of a simple linear regression. Let β0 = 2 and β1 = 1 and assume the following relationship holds: y = β0 + β1x + epsilon, where x ∈ [0, 10] and epsilon ∈ N(0, σ2) with σ2 = 3. For each combination of n and m wheren ∈ {20, 100} and m ∈ {10, 500}, do the following:

Create a vector of n evenly spaced values for x in the range [0,10], then generate n random observations of y according to the model above

```
Xvector <- function(n)
{
    return (seq(0, 10, length.out = n))
}

Xvector(5)
```

```
## [1]  0.0  2.5  5.0  7.5 10.0
```

```
Yvector <- function(x)
{
  epsilon = rnorm(length(x), mean = 0, sd = sqrt(3))
  y = 2 + 1*x + epsilon
  return (y)
}
Yvector(Xvector(5))
```

```
## [1]  2.901668  6.362925  7.181206 10.185965  9.953682
```

Estimate the values of β0 and β1 using the least squares method (lm function).

```
CreateDataFrame <- function(n)
{
  df <- data.frame(Xvector(n),Yvector(Xvector(n)))
  names(df) <- c("X","Y")
  return (df)
}

GetLm <- function(n)
{
  return (model = lm(data = CreateDataFrame(n),formula =Y~X))
}

GetLm(5)$coefficients
```

```
## (Intercept)           X
##    2.445437    0.935845
```

Repeat steps a) and b) m times.

```
GetCoefficientsDf <- function(m,n)
{
  if(m<1){
    return (NULL)
  }
  df <- data.frame(X= numeric(0), Y= numeric(0))
  for(model in 1:m)
    {
    de <- GetLm(n)$coefficients
    df <- rbind(df, de)
  }
  names(df)<-c("β0","β1")
  return (df)
}

GetCoefficientsDf(2,12)
```

```
##   <U+03B2>0 <U+03B2>1
## 1 1.1098263 1.0036383
## 2 0.6133471 0.9708139
```

Calculate the mean and variance of the estimates for β0 and β1 over the m simulations.

```
GetMeanSD <-function(m,n)
{
  df = GetCoefficientsDf(m,n)
  sd = apply(df, 2, sd)
  mean = apply(df, 2, mean)
  var = sd*sd
  return(rbind(mean,var))
}

GetMeanSD(100,100)
```

```
##      <U+03B2>0   <U+03B2>1
## mean 1.9852021 0.997909728
## var  0.1012728 0.003375533
```

Create a table summarising the results that includes the columns: n, m, beta0_mean, beta1_mean, beta0_var and beta1_var. Explain your findings regarding the effect of the size of n and the size of m on the results.

```
CreateSummarisingTable <- function(m,n)
{
  exData = GetMeanSD(m,n)
  df =
data.frame(n,m,exData["mean","β0"],exData["mean","β1"],exData["var","β0"],exData["var","β1"])
  names(df)<-c("n", "m", "beta0_mean","beta1_mean", "beta0_var", "beta1_var")
  return(df)
}

CreateSummarisingTable(100,100)
```

```
##     n   m beta0_mean beta1_mean beta0_var  beta1_var
## 1 100 100   1.993026   1.004109 0.1308815 0.00427993
```

the best estimation would be β0=2 & β1=1. we can see that the corolation is positive-> for n->inf m->inf we would get β0=2 & β1=1. that is becouse in both cases epsilon would be 0 and the simulation would have the same formula as the linear model.

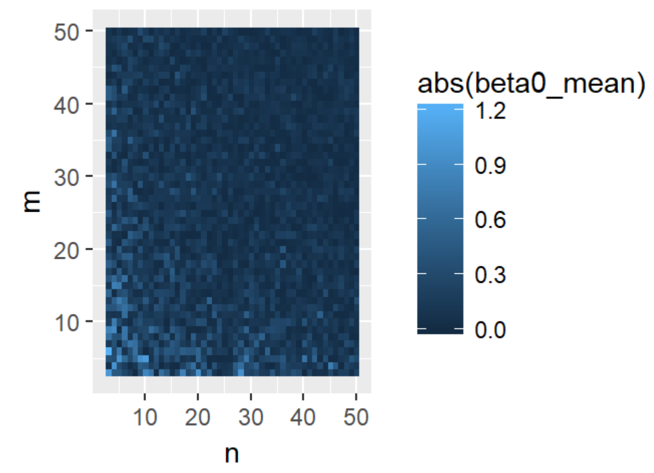
```
library(ggplot2)
```

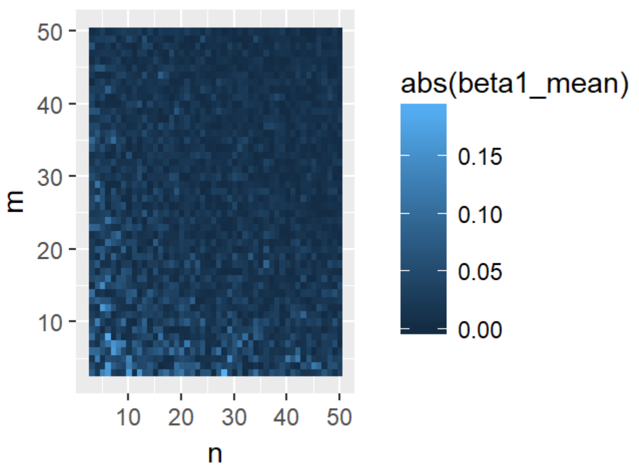
```
## Warning: package 'ggplot2' was built under R version 3.4.1
```

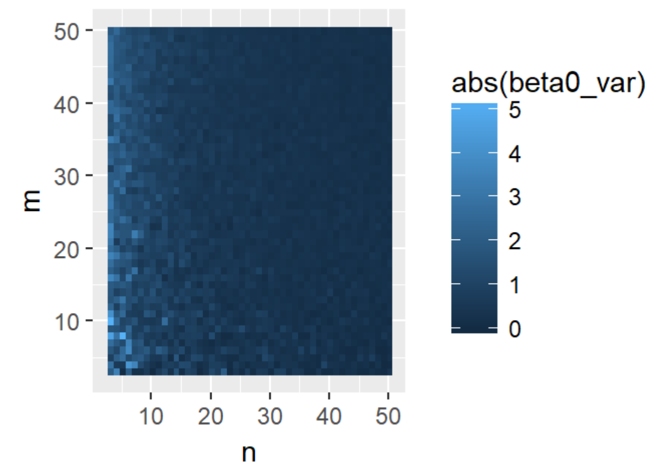
```
library(gridExtra)
```

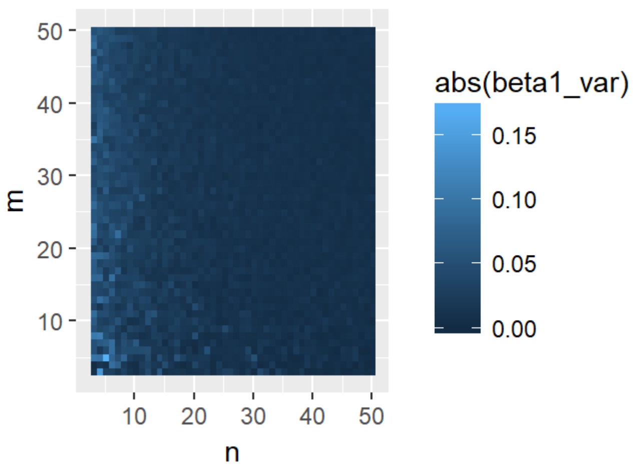
```
## Warning: package 'gridExtra' was built under R version 3.4.1
```

```
seeResult <- function(Maxm,Maxn)
  {
  df <- data.frame(n= Maxm,m= Maxm,beta0_mean= 0,beta1_mean= 0, beta0_var= 0,  beta1_var= 0)
    for(m in 3:Maxm){
    for(n in 3:Maxn)
    {
      addone = CreateSummarisingTable(m,n)
      addone$beta0_mean = addone$beta0_mean-2
      addone$beta1_mean = addone$beta1_mean-1
      df=rbind(df,addone)
     }
  }
  return (df)
}

df<-seeResult(50,50)
beta0_mean<-ggplot(data =df)+
  geom_raster(mapping = aes(n,m,fill= abs(beta0_mean)), data = df)
beta1_mean<-ggplot(data =df)+
  geom_raster(mapping = aes(n,m,fill= abs(beta1_mean)), data = df)
beta0_var<-ggplot(data =df)+
  geom_raster(mapping = aes(n,m,fill= abs(beta0_var)), data = df)
beta1_var<-ggplot(data =df)+
  geom_raster(mapping = aes(n,m,fill= abs(beta1_var)), data = df)
grid.arrange(beta0_mean, beta1_mean,beta0_var,beta1_var)
```









answer

we can see in the figure 4 models for each estimation value with incrementing n and m.

in mean estimators the value in each pixel represented as the absolute distance from the real mean. where in sd estimators it is represented as the same value.

from this figure we can see the gradient of change in value: in all 4 cases the gradient direction is very similar to the ‘n’ axis, meaning ‘n’ margin is better then ‘m’,

we can also see that the mean estimators gradient direction gets more influence from ‘m’ meaning that the mean estimation relay more on the number of iteration averaged than the sd estimator.

Execrise #2:

In this exercise you will implement the K-nearest-neighbor (KNN) method for regression. Let d(a, b) = qPpi=1 (ai − bi)2 be the Euclidean distance between points a and b. Assume we have a training set of x1, …, xn points (observations with p ≥ 1 dimensions) and outcome values y1, …, yn, and we are given x0 as a new test point. Define di(x0) = d(x0, xi) as the distance between x0 and each point xi. Let d(i)(x0) be the distance between x0 and the i’th nearest point to x0 (i.e., d(1)(x0) is the distance between x0 and the nearest point, d(2)(x0) is the distance between x0 and the second nearest point. . .). The set of K-nearest neighbors of x0 is given by Nk(x0) = {xi: di(x0) ≤ d(K)(x0)}. The KNN estimator of y0 is given by yˆ0 = 1KPi:xi∈Nk(x0) yi.

Create a function in R that calculates the KNN estimator at a point x0. The function should get as input the points (matrix of size n × p), the outcomes (n × 1), the new point x0 (1 × p) and the number K. The output should be the estimate for y0.

```r
calcDistance <- function(a,b)
{
  if(length(a)!=length(b))
  {
    return (NULL)
  }
  vac = (a-b)^2
  value <- sum(vac)
  return (sqrt(value))
}

GetDistanceMatrix <- function(points,newpoint,outcomes)
{
  numberOfRow = nrow(points)
  returnValue = matrix(0, nrow = numberOfRow, ncol = 1)
  for(index in 1:numberOfRow)
  {
    ans = calcDistance(points[index, ],newpoint)
    returnValue[index,1] =ans
    if(is.null(ans))
    {
      print("error while calc distance")
    }
  }
  return (data.frame(returnValue,outcomes))
}


KNNestimator <- function(points,outcomes,newpoint,knumber)
{
  data <- GetDistanceMatrix(points,newpoint,outcomes)
  head <- head(data[order(data[,1]),], knumber)

  if(is.character(outcomes))
  {
    slice(
      arrange(
        count(head, vars = outcomes)
        ,desc(n))
      , 1 )  -> firstk
    return (firstk[1,1]$vars)
  }
  else if(is.double(outcomes))
  {
    return(sum(head$outcomes)/knumber)
  }
  else
  {
    return("Dont support outcome type")
  }
}


sizeFactor = 50
k=50
propvec = c(0.25, 0.25, 0.25, 0.25)
points=matrix(rexp(sizeFactor*sizeFactor), sizeFactor)
```

```
newpoint = rexp(sizeFactor)
characterOutcomes = sample( LETTERS[1:4], sizeFactor, replace=TRUE, prob=propvec)
doubleOutcomes = rexp(sizeFactor)

(KNNestimator(points,characterOutcomes,newpoint,k))
```

```
## [1] A
## Levels: A B C D
```

```
(KNNestimator(points,doubleOutcomes,newpoint,k))
```

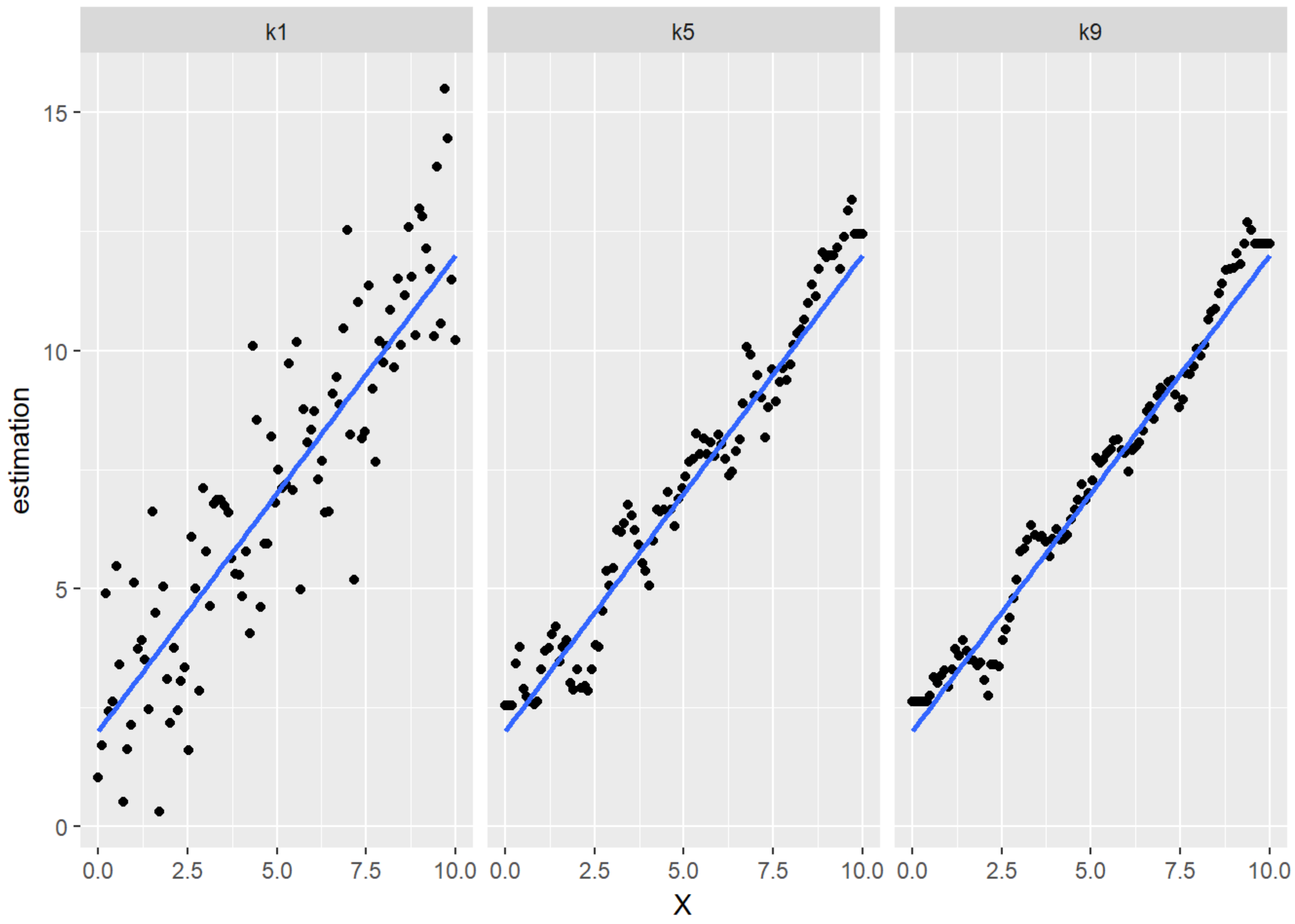
```
## [1] 0.8314869
```

Generate x and y using the setup from question 1 with n = 100. Calculate the KNN estimator with K = 1, 5, 9 for the points x0 = x1, …, xn (i.e., for each point xi, estimate yi using all the points in the training set). For each K, create a ggplot with the true function (yi without the noise term - drawn using a black line), the observations (yi -drawn using black circles) and the KNN estimators (yˆi - drawn using a red line).

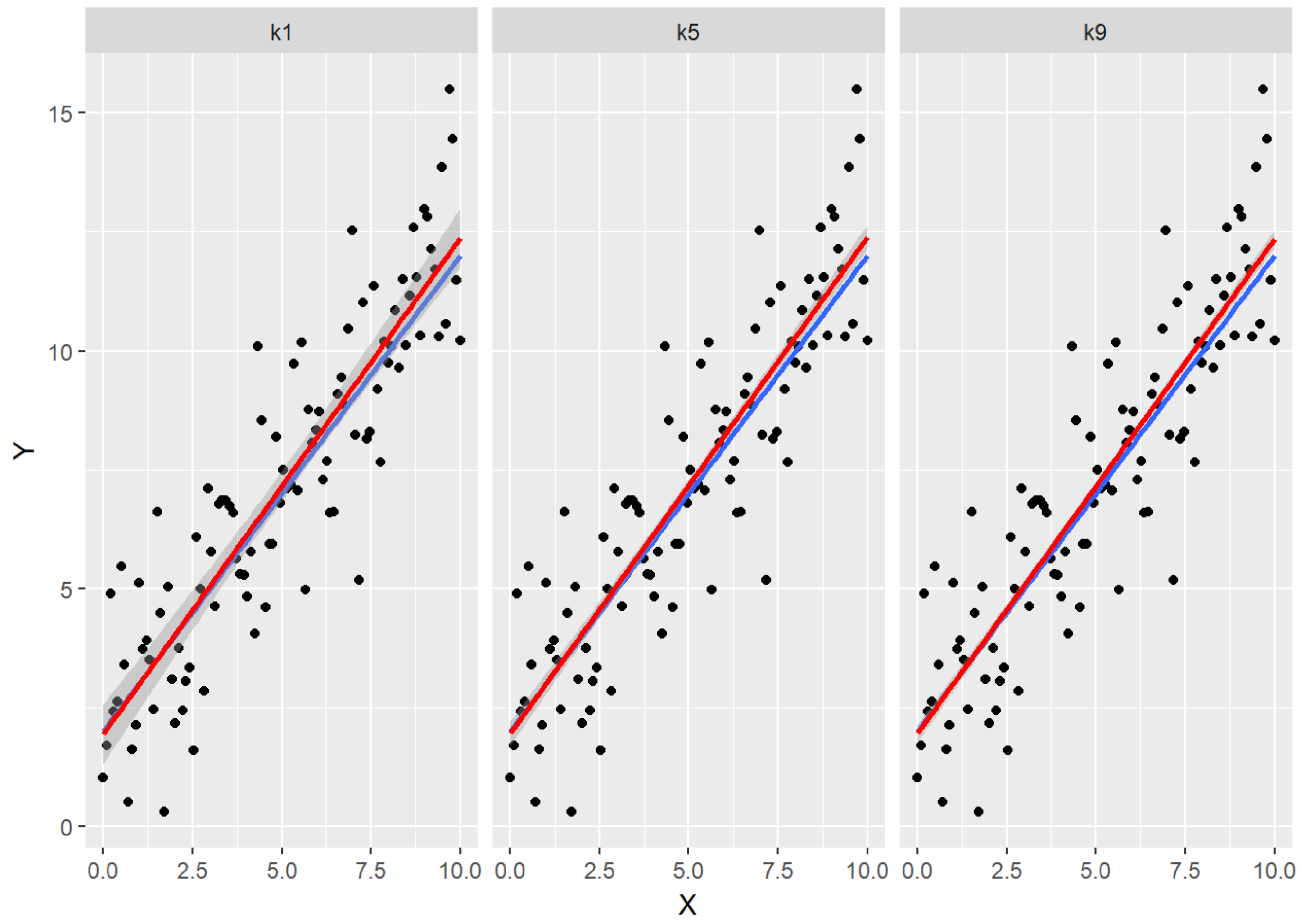
```
Xvector <- function(n)
{
  return (seq(0, 10, length.out = n))
}

Yvector <- function(x)
{
  set.seed(123)
  epsilon = rnorm(n=length(x), mean = 0, sd = sqrt(3))
  y = 2 + 1*x + epsilon
  return (y)
}


n=100

xdata = as.data.frame(Xvector(n))
ydata = Yvector(Xvector(n))
ytrend =  2 + 1*Xvector(n)

returnValue <- data.frame(x= xdata, y= ydata,
ytrend,est_y_k1=numeric(n),est_y_k5=numeric(n),est_y_k9=numeric(n))
for(row in 1:nrow(returnValue))
{
  x0<-  returnValue[row,'Xvector.n.']
  returnValue[row,'est_y_k1']<-KNNestimator(xdata,ydata,x0,1)
  returnValue[row,'est_y_k5']<-KNNestimator(xdata,ydata,x0,5)
  returnValue[row,'est_y_k9']<-KNNestimator(xdata,ydata,x0,9)
}
colnames(returnValue) <- c("X", "Y", "Trend", "k1", "k5", "k9")
head(returnValue)
```

```
##           X        Y     Trend       k1       k5       k9
## 1 0.0000000 1.029228 2.000000 1.029228 2.537294 2.633425
## 2 0.1010101 1.702331 2.101010 1.702331 2.537294 2.633425
## 3 0.2020202 4.901782 2.202020 4.901782 2.537294 2.633425
## 4 0.3030303 2.425154 2.303030 2.425154 3.426574 2.633425
## 5 0.4040404 2.627973 2.404040 2.627973 3.766986 2.633425
## 6 0.5050505 5.475630 2.505051 5.475630 2.889814 2.756531
```

```
gatheredValues = gather(data = returnValue,key = K, value = estimation,... = k1:k9,factor_key
 = TRUE)

ggplot(data =gatheredValues)+
  geom_point(mapping = aes(X,estimation))+
  geom_smooth(mapping = aes(x = gatheredValues$X, y = gatheredValues$Trend), method='lm')+
  facet_wrap( ~K)
```



```
ggplot(data =gatheredValues)+
  geom_point(mapping = aes(X,Y))+
  geom_smooth(mapping = aes(x = gatheredValues$X, y = gatheredValues$Trend), method='lm')+
  geom_smooth(mapping = aes(x = gatheredValues$X, y = gatheredValues$estimation),
method='lm',color='red')+
  facet_wrap( ~K)
```



c. Explain how would you go about selecting a value for K to use for prediction using the training set.

->Select the k that minimize the error can use cross validation to get the optimal value.

Execrise #3:

3. In this exercise you will employ logistic regression with the '2016 World Health Report' data set, to examine how well can we predict if a country is European or African based on the given predictors.

a. Using the 'dplyr' package or any other programming method you want to apply, create a subset of the '2016 World Health Report' data set, that includes only European or African countries (50 European countries and 43 African countries). The new data set will have a new field called 'Region2' that will contain either 'Europe' or 'Africa', as well as a field called 'IsEuropean' which is set to true for European countries and false otherwise.

```
world <- read.csv('./2016 World Happiness Report.csv',header=T)
attach(world)
summary(world$Region)
```

```
##       Australia and New Zealand      Central and Eastern Europe 
##                               2                              29 
##                    Eastern Asia     Latin America and Caribbean 
##                               6                              24 
## Middle East and Northern Africa                   North America 
##                              19                               2 
##              Southeastern Asia                  Southern Asia 
##                               9                               7 
##              Sub-Saharan Africa                  Western Europe 
##                              38                              21
```

```
EuroGroup =c("Central and Eastern Europe","Western Europe")
AfroGroup =c("Middle East and Northern Africa","Sub-Saharan Africa")

df2 <- filter(.data = world,Region %in%  c(EuroGroup, AfroGroup) )  %>%
  mutate(Region2=if_else(Region %in%  EuroGroup, "Europe", "Africa", missing = NULL),
         IsEuropean=if_else(Region %in%  EuroGroup, TRUE, FALSE, missing = NULL))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.1
```

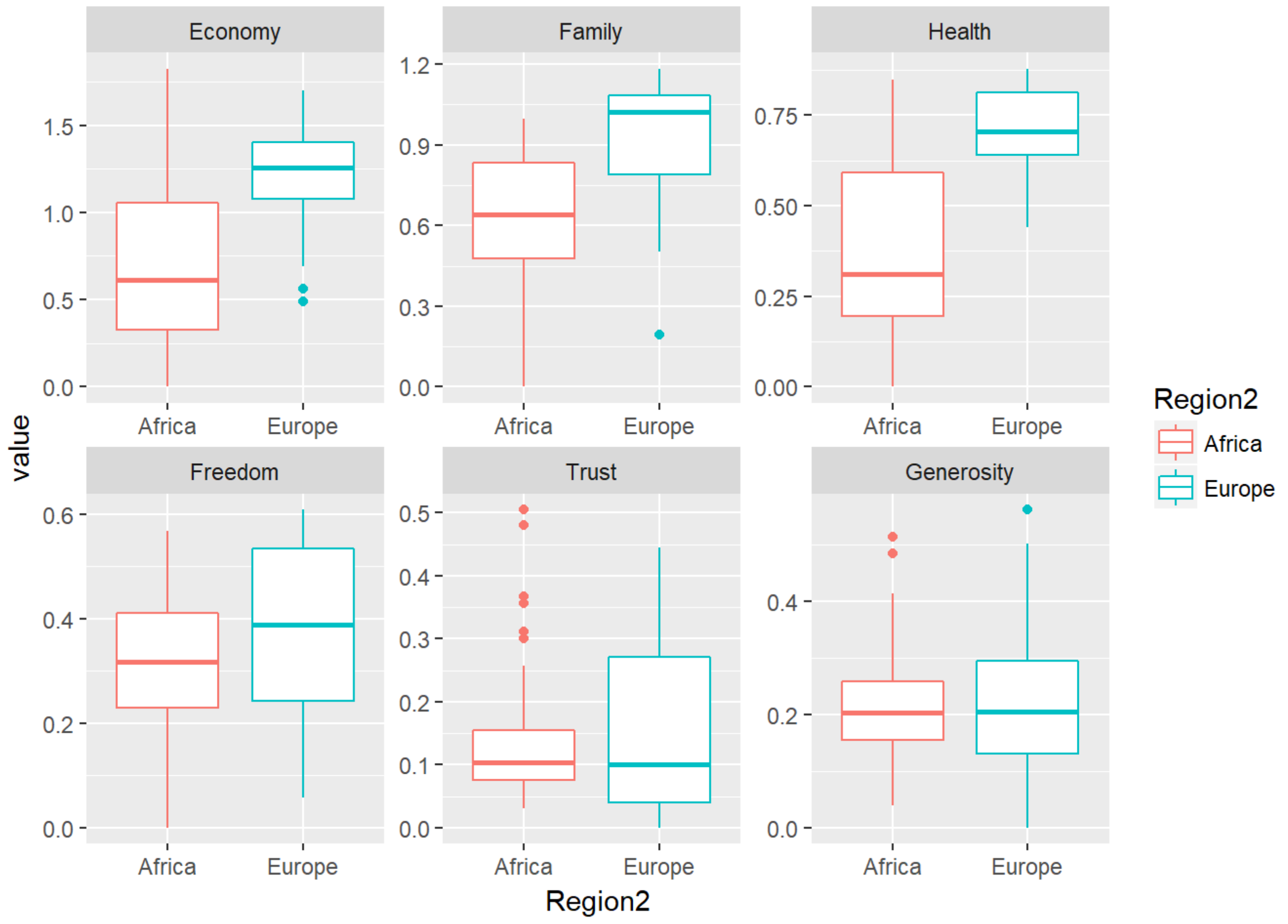
```
df2$Region2 <- as.factor(df2$Region2)

summary(df2)
```

```
##        Country                                 Region     Happiness    
##  Albania   :  1   Sub-Saharan Africa             :38   Min.   :2.905  
##  Algeria   :  1   Central and Eastern Europe     :29   1st Qu.:4.228  
##  Angola    :  1   Western Europe                 :21   Median :5.145  
##  Armenia   :  1   Middle East and Northern Africa:19   Mean   :5.193  
##  Austria   :  1   Australia and New Zealand      : 0   3rd Qu.:5.948  
##  Azerbaijan:  1   Eastern Asia                   : 0   Max.   :7.526  
##  (Other)   :101   (Other)                        : 0                  
##     Economy           Family           Health          Freedom      
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.5578   1st Qu.:0.6251   1st Qu.:0.3006   1st Qu.:0.2347  
##  Median :1.0344   Median :0.8068   Median :0.5947   Median :0.3602  
##  Mean   :0.9328   Mean   :0.7725   Mean   :0.5279   Mean   :0.3422  
##  3rd Qu.:1.2896   3rd Qu.:1.0065   3rd Qu.:0.7236   3rd Qu.:0.4387  
##  Max.   :1.8243   Max.   :1.1833   Max.   :0.8790   Max.   :0.6085  
##                                                                     
##      Trust          Generosity       Region2   IsEuropean     
##  Min.   :0.0000   Min.   :0.0000   Africa:57   Mode :logical  
##  1st Qu.:0.0540   1st Qu.:0.1389   Europe:50   FALSE:57       
##  Median :0.1039   Median :0.2036               TRUE :50       
##  Mean   :0.1409   Mean   :0.2197                              
##  3rd Qu.:0.1786   3rd Qu.:0.2774                              
##  Max.   :0.5052   Max.   :0.5624                              
## 
```

is more then 43 since it is mixed with middel east countries but i assume and hope this is not the point of this execrise

b. Using ‘ggplot2’ and with the help of ‘tidyr’, create a plot similar to Figure1 below. Based on this figure, which predictors could be helpful in predicting if a country is from Europe or Africa?

```
world_long = gather(data = df2,key = sector, value = value,... = Economy:Generosity,factor_ke
y = TRUE)

ggplot(data =world_long)+
  geom_boxplot(mapping = aes(x=Region2,y = value,color= Region2))+
  facet_wrap( ~sector, scales = "free")
```



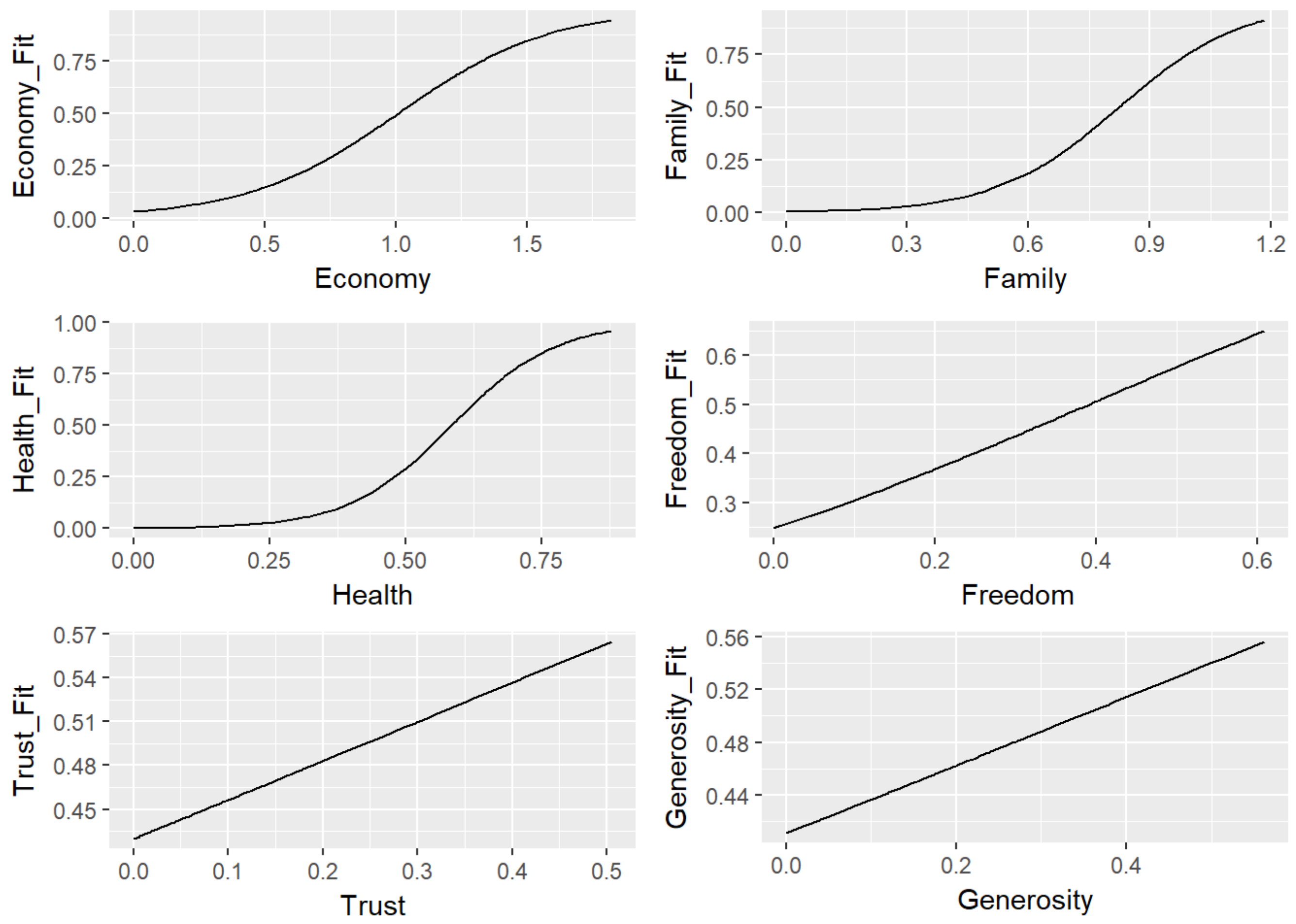
best proprties base on this plot are health, economy, and family.

c) Using glm, fit a simple logistic regression to IsEuropean using each of the six predictors shown in Figure1. According to the fits, which predictor by itself is better in distinguishing between European and African countries? Create a ggplot showing the probability of a country being European according to the value of this best predictor.

```
library(ggplot2)
attach(df2)
```

```
## The following objects are masked from world:
##
##     Country, Economy, Family, Freedom, Generosity, Happiness,
##     Health, Region, Trust
```

```
logm.Economy<-glm(formula = IsEuropean ~ Economy, family =binomial , data = df2)
logm.Family<-glm(formula = IsEuropean ~ Family, family =binomial , data = df2)
logm.Health<-glm(formula = IsEuropean ~ Health, family =binomial , data = df2)
logm.Freedom<-glm(formula = IsEuropean ~ Freedom, family =binomial , data = df2)
logm.Trust<-glm(formula = IsEuropean ~ Trust, family =binomial , data = df2)
logm.Generosity<-glm(formula = IsEuropean ~ Generosity, family =binomial , data = df2)
Economy_Fit=fitted(logm.Economy)
Family_Fit=fitted(logm.Family)
Health_Fit=fitted(logm.Health)
Freedom_Fit=fitted(logm.Freedom)
Trust_Fit=fitted(logm.Trust)
Generosity_Fit=fitted(logm.Generosity)
g1 <-ggplot() +  geom_line(mapping=aes(x=Economy,y=Economy_Fit))
g2 <-ggplot() +  geom_line(mapping=aes(x=Family,y=Family_Fit))
g3 <-ggplot() +  geom_line(mapping=aes(x=Health,y=Health_Fit))
g4 <-ggplot() +  geom_line(mapping=aes(x=Freedom,y=Freedom_Fit))
g5 <-ggplot() +  geom_line(mapping=aes(x=Trust,y=Trust_Fit))
g6 <-ggplot() +  geom_line(mapping=aes(x=Generosity,y=Generosity_Fit))
grid.arrange(g1,g2,g3,g4,g5,g6)
```



we can see that health hold the best separator between 0 probability to be European and 1 - it is the most similar to zigmond function d) Based on the fit using the best predictor from c), create a confusion table of actual region vs. predicted region, using each of the following threshold values for p(IsEuropean=Yes): 1/4, 1/2, 3/4. Report the sensitivity and specificity of the classifier in each case. Which of these classifiers would you use? Specify if it depends on anything.

```
confusion.glm <- function(data, model,tresh) {
  prediction <- ifelse(predict(model, data, type='response') > tresh, TRUE, FALSE)
  confusion  <- table(prediction, as.logical(model$y))
  confusion  <- cbind(confusion, c(1 - confusion[1,1]/(confusion[1,1]+confusion[2,1]), 1 - co
nfusion[2,2]/(confusion[2,2]+confusion[1,2])))
  confusion  <- as.data.frame(confusion)
  names(confusion) <- c('FALSE', 'TRUE', 'class.error')
  confusion
}
confusion.glm(df2,logm.Health,0.25)
```

```
##       FALSE TRUE class.error
## FALSE    38    1   0.3333333
## TRUE     19   49   0.0200000
```

```
confusion.glm(df2,logm.Health,0.5)
```

```
##       FALSE TRUE class.error
## FALSE    42    8   0.2631579
## TRUE     15   42   0.1600000
```

```
confusion.glm(df2,logm.Health,0.75)
```

```
##       FALSE TRUE class.error
## FALSE    53   22  0.07017544
## TRUE      4   28  0.44000000
```

Execrise #4:

4. In this exercise you will employ the ‘caret’ package wrapper for randomForest with the ‘2016 World Health Report’ data set, in order to fit and tune a model for predicting the Happiness score of a country.

a. Use caret::createDataPartition to create training and test sets, with the training set containing 70% of the data.

```
world <- read.csv('./2016 World Happiness Report.csv',header=T)
attach(world)
```

```
## The following objects are masked from df2:
##
##     Country, Economy, Family, Freedom, Generosity, Happiness,
##     Health, Region, Trust
```

```
## The following objects are masked from world (pos = 5):
##
##     Country, Economy, Family, Freedom, Generosity, Happiness,
##     Health, Region, Trust
```

```
#install.packages("caret")
#install.packages("caret", dependencies = c("Depends", "Suggests"))

library("caret")
```

```
## Warning: package 'caret' was built under R version 3.4.1
```

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

```
train<-createDataPartition(y=Happiness,  p = 0.7, list = FALSE)
world.train <- world[train,]
world.test <- world[-train,]
Happiness.test <- world$Happiness[-train]
```

b. Explain what is the meaning of the tuning parameter ‘mtry’ of randomForest. What is the maximum number it can take in our case, assuming we want to fit Happiness using all the predictors in the data set (not including the field ‘Country’)?

mtry is the number of variables randomly sampled as candidates at each split, the maximum number it can take is bound by the number of variables in the model, as it specifies the size of the variable subset that is randomly picked for each random forest iteration. the maximun number would be all distinct factors in each factor feture pluse the number of numric fetures:

```
numricValues = table(sapply(world, is.numeric))["TRUE"]
factorValues = sapply(world[,sapply(world, is.factor)], nlevels)
(maxmtry = sum(factorValues,numricValues))
```

```
## [1] 174
```

maximum value for mtry =174

c. Use caret::train with method=‘rf’ to fit the model to the training set. Set that the method will search for the best possible value for ‘mtry’ out of all possible values, using root-mean-squre error (RMSE) as the metric for selecting the best value. Set the resampling method to be used to 10-fold cross-validation. Set the number of trees to be grown to 500. Make sure that the importance of the predictors will be assessed. Using the obtained fit, plot the cross-validation RMSE as a function of mtry and the cross-validation Rsquared as a function of mtry.

What is the selected value for ‘mtry’ according to the fit? What would have been the selected value for ‘mtry’ if we used Rsquared as the metric for selection?

```
#install.packages("randomForest")
#library(plyr)
set.seed(123)
Happiness.rfFit <- train(Happiness~.,world.train,
                      method='rf',
                      metric='RMSE',
                      trControl=trainControl(method='cv',number=10),
                      tuneGrid=data.frame(mtry=3:150),
                      ntree=500,
                      importance=TRUE)
```

```
## Warning: package 'randomForest' was built under R version 3.4.1
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

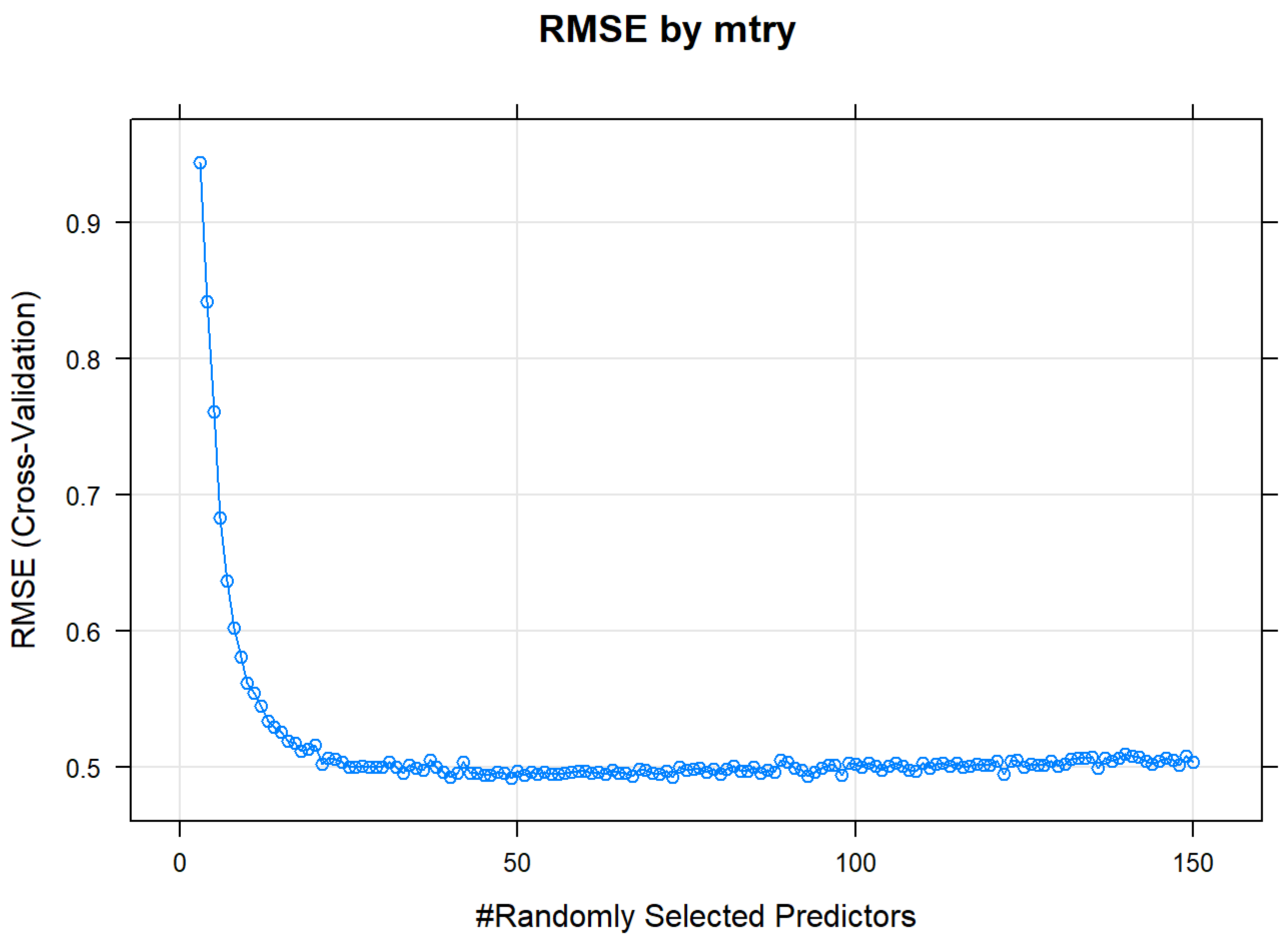
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

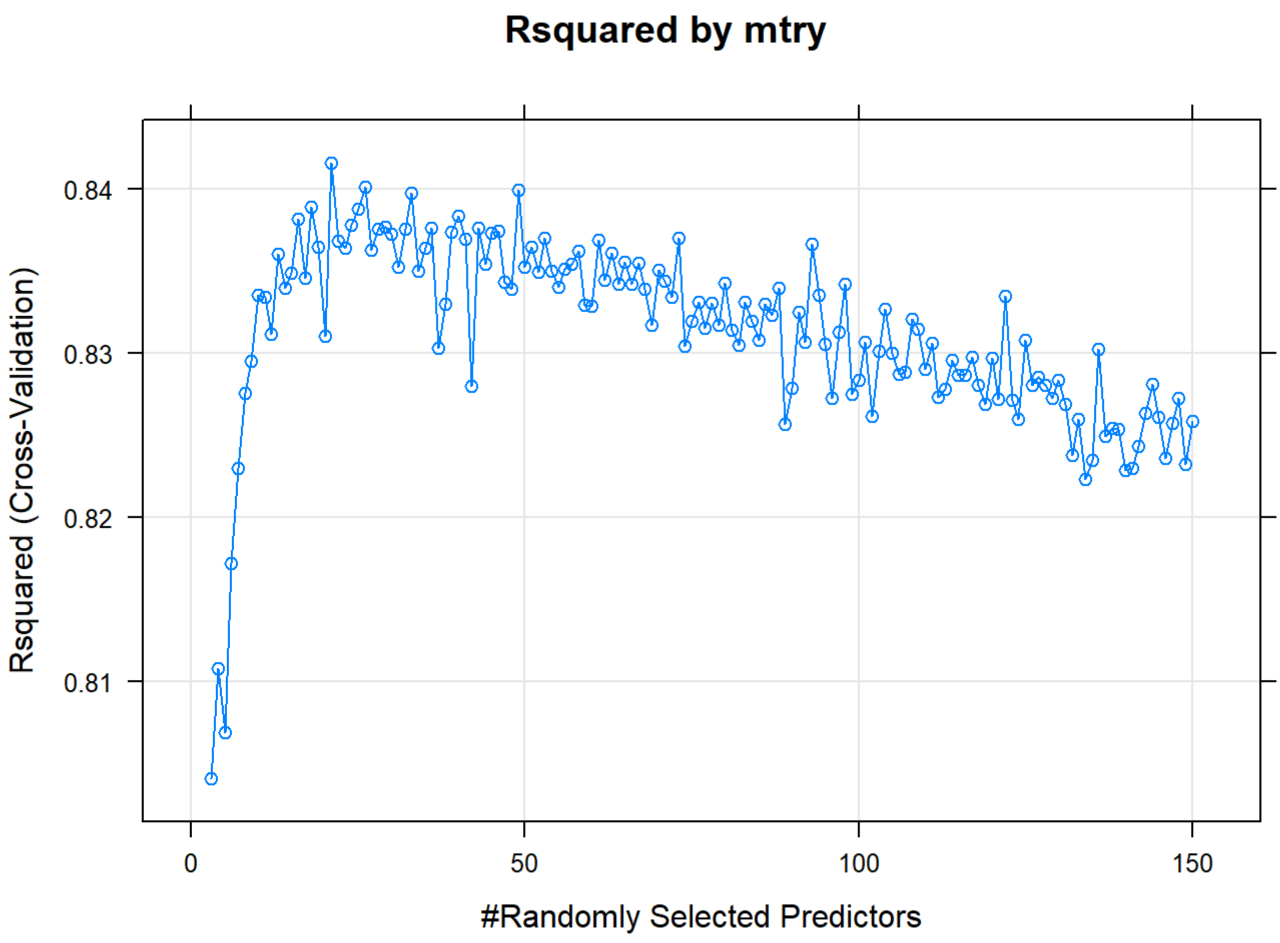
```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
plot(Happiness.rfFit, metric='RMSE',main="RMSE by mtry")
```



```
plot(Happiness.rfFit, metric='Rsquared',main="Rsquared by mtry")
```



```
getTrainPerf(Happiness.rfFit)
```

```
##   TrainRMSE TrainRsquared  TrainMAE method
## 1 0.4918707     0.8399018 0.3980317     rf
```

```
(winning.RMSE = Happiness.rfFit$results[which.min(Happiness.rfFit$results[, "RMSE"]), ])
```

```
##    mtry      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 47   49 0.4918707 0.8399018 0.3980317 0.09740359 0.05667068 0.07089441
```

```
(winning.Rsquared = Happiness.rfFit$results[which.max(Happiness.rfFit$results[, "Rsquared"]),
 ])
```

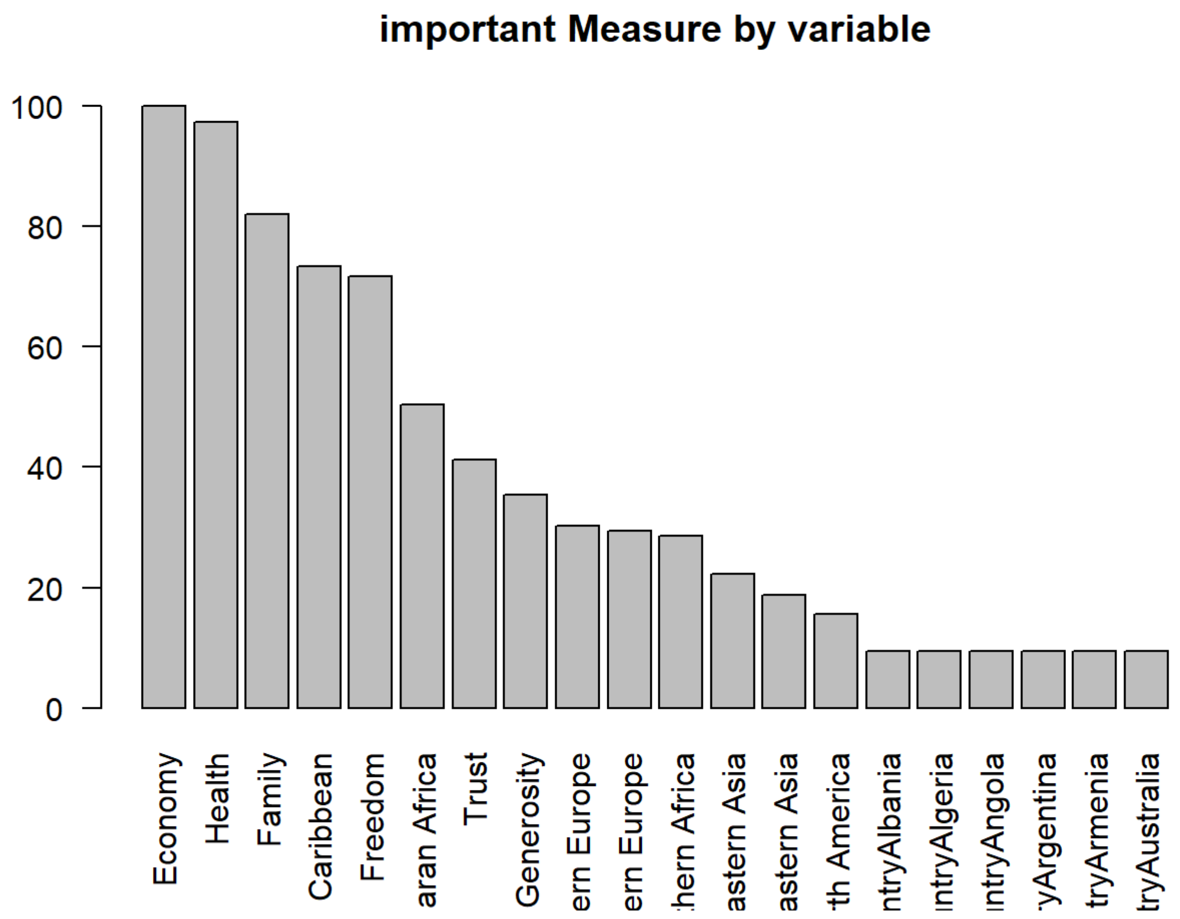
```
##    mtry      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 19   21 0.5021744 0.8415588 0.4081469 0.09216605 0.05829391 0.06812168
```

```
mtryw = winning.Rsquared$mtry
```

the wining mtry for Rsquared is mtry= 21

d. Using the obtained fit, plot the calculated variable importance. What are the top-3 predictors effecting Happiness according to this plot?

```
ImpMeasure<-data.frame(varImp(Happiness.rfFit)$importance)
ImpMeasure$Vars<-row.names(ImpMeasure)
ImpMeasure<-ImpMeasure[order(-ImpMeasure$Overall),][1:20,]

barplot(ImpMeasure$Overall, names.arg=ImpMeasure$Vars ,main="important Measure by variable",l
as=2)
```



```
top = head(ImpMeasure,3)$Vars
```

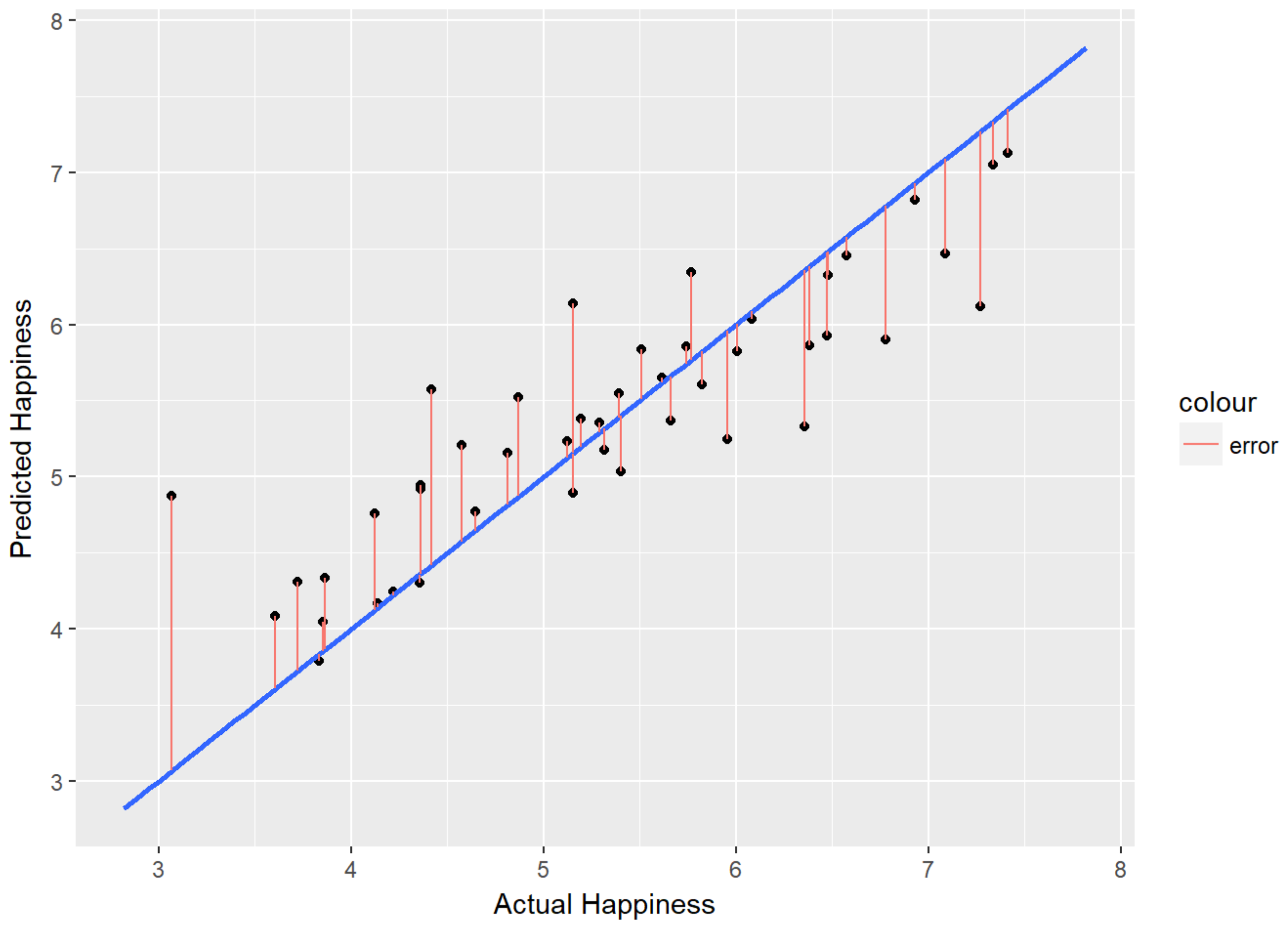
the top-3 are :Economy, Health, Family

e. Use the model fit to estimate the Happiness score for the test set. Calculate the RMSE for these predictions. Create a ggplot with the actual Happiness score vs. the predicted Happiness score.

```
Happiness.rf <- predict(Happiness.rfFit, newdata=world.test)
(test.mse.rf <- mean((Happiness.rf - Happiness.test)^2))
```

```
## [1] 0.3144871
```

```
range = (min(Happiness.test)-0.25):(max(Happiness.test)+0.75)
ggplot()+
  geom_point(mapping = aes(x=Happiness.test,y = Happiness.rf))+
  ylab("Predicted Happiness")+
  xlab("Actual Happiness")+
  geom_smooth(mapping = aes(x = range, y = range), method='lm')+
  geom_segment(aes(x=Happiness.test, xend=Happiness.test, y=Happiness.rf,
yend=Happiness.test, color="error"))
```


Predicted Happiness
Actual Happiness
3
4
5
6
7
8
colour
error