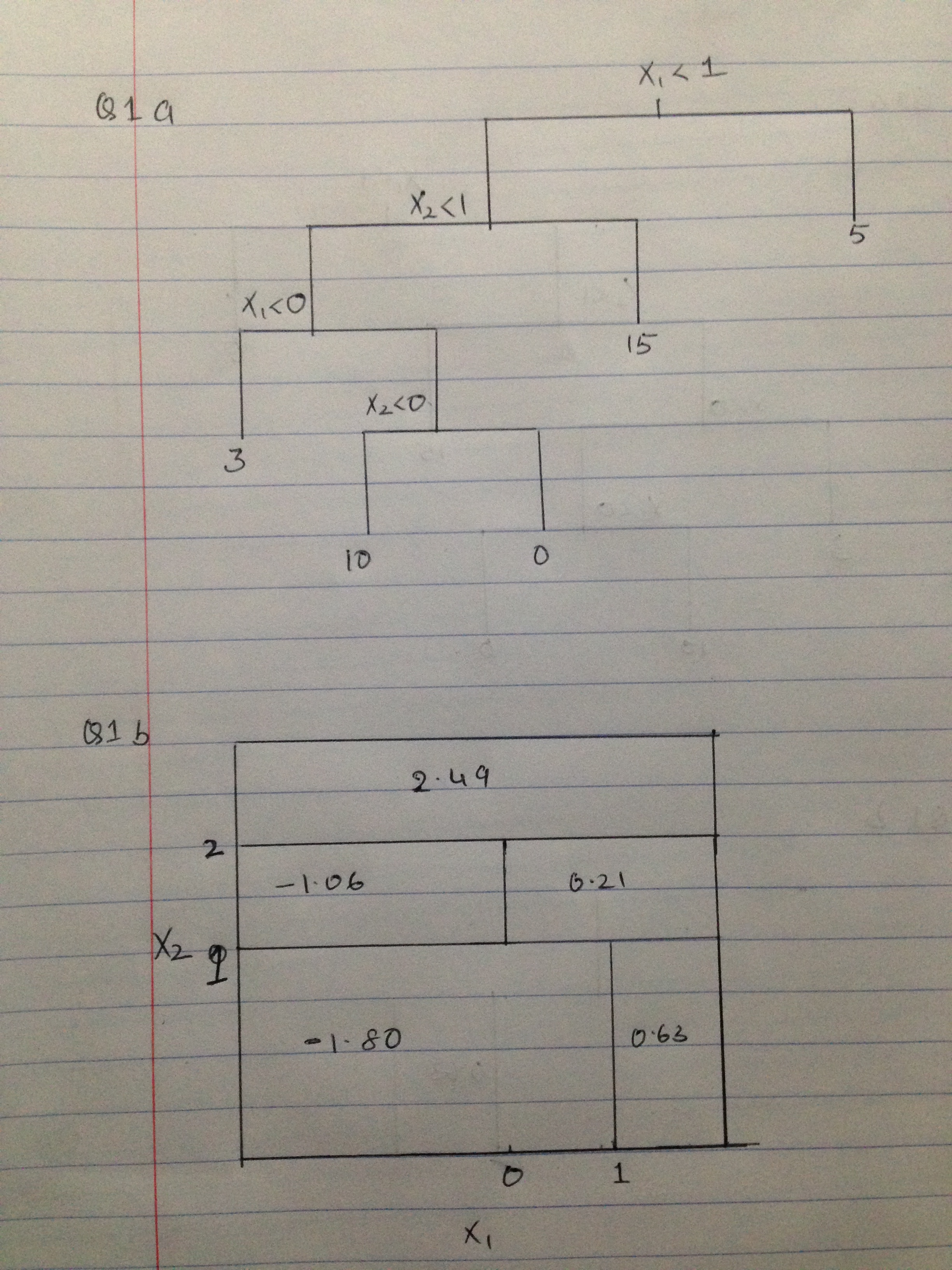
HW4

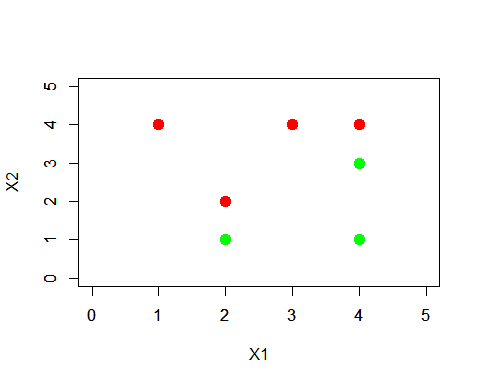
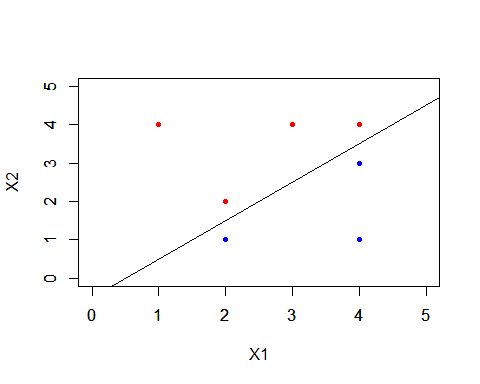
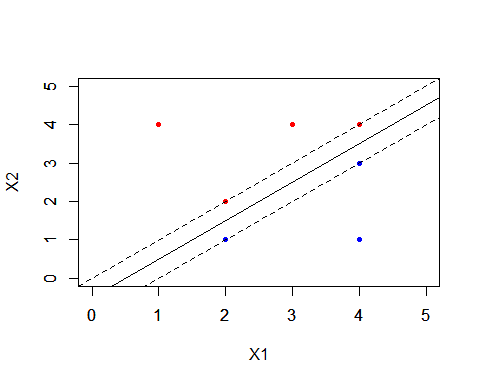
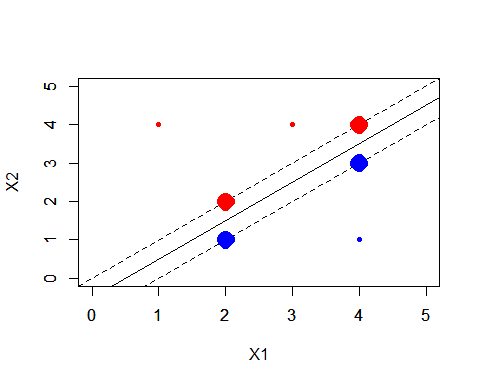
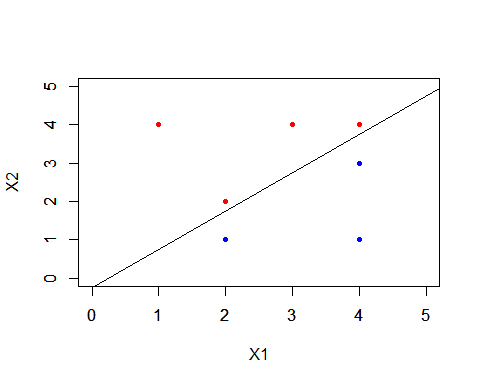
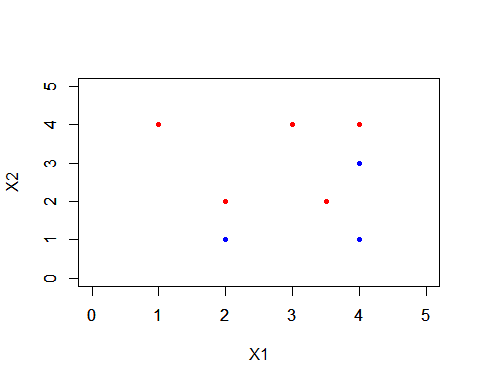
Anish Mohan

March 9, 2016

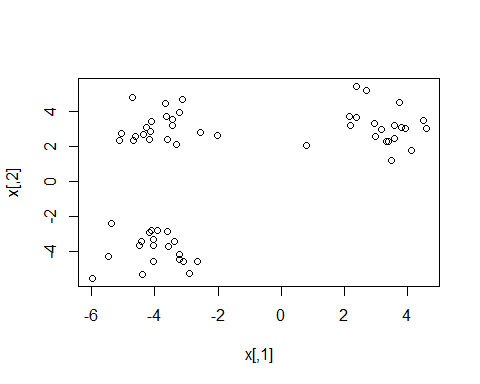
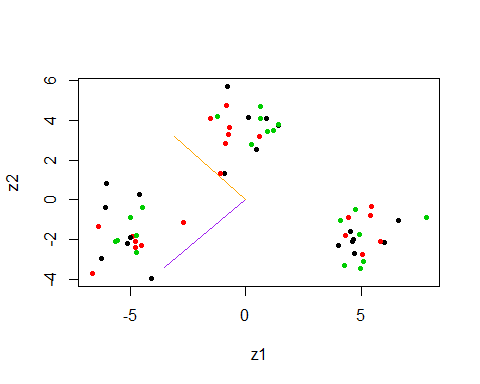
1. Q1.

* 

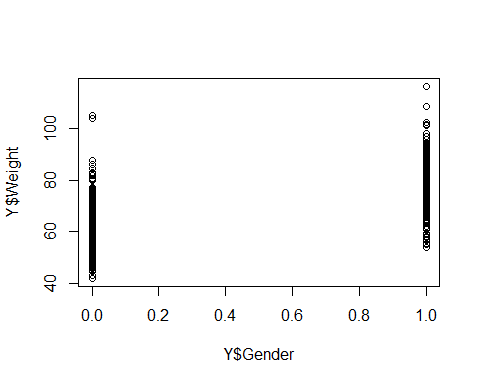
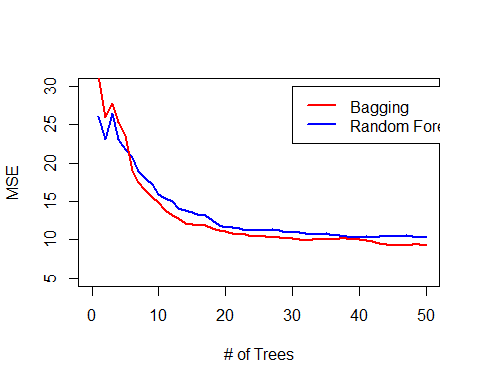
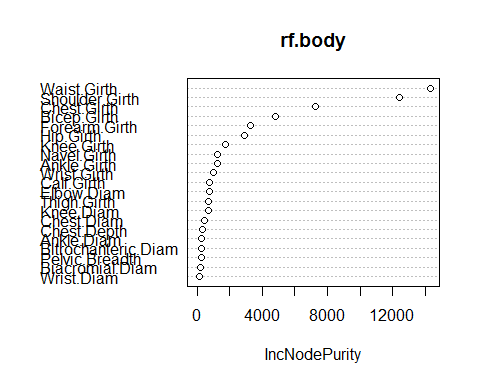
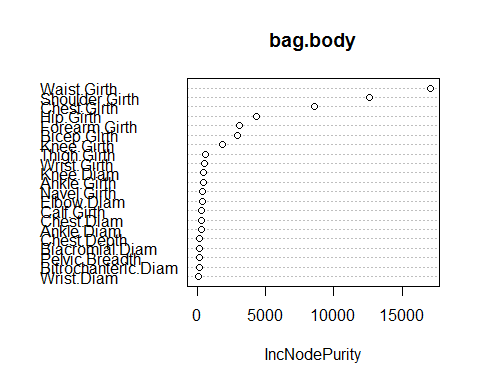
1. Q2.

* 2a.
* X1op=c(3,2,4,1,2,4,4)  
  X2op=c(4,2,4,4,1,3,1)  
  Yop=c("Red","Red", "Red", "Red", "Blue", "Blue","Blue")  
    
  plot(X1op[5:7],X2op[5:7],col="green", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",lwd=6)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", lwd=6)
* 
* par(new=F)
* 2b.
* X1op=c(3,2,4,1,2,4,4)  
  X2op=c(4,2,4,4,1,3,1)  
  Yop=c("Red","Red", "Red", "Red", "Blue", "Blue","Blue")  
    
  plot(X1op[5:7],X2op[5:7],col="blue", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",pch=20)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", pch=20)  
  par(new=F)  
  abline(-0.5,1)
* 
  + Equation of the hyperplane is
* 2c.
  + Eqn of hyperplane is , so subbing values
  + Blue (2,1)= 2(2)-2(1)-1= 4-3=1
  + Red (2,2)=2(2)-2(2)-1=4-4-1=-1
  + -> == Blue
  + -> == Red
* 2d
  + margin is the perpendicular distance from a point to line e.g red point at (2,2) intersects the line at (2.25, 1.75) distance between them is about 0.3
* X1op=c(3,2,4,1,2,4,4)  
  X2op=c(4,2,4,4,1,3,1)  
  Yop=c("Red","Red", "Red", "Red", "Blue", "Blue","Blue")  
    
  plot(X1op[5:7],X2op[5:7],col="blue", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",pch=20)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", pch=20)  
  par(new=F)  
  abline(-0.5,1)  
  abline(-1,1,lty=2)  
  abline(0,1,lty=2)
* 
* 2e
* plot(X1op[5:7],X2op[5:7],col="blue", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",pch=20)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", pch=20)  
  par(new=F)  
  abline(-0.5,1)  
  abline(-1,1,lty=2)  
  abline(0,1,lty=2)  
    
  points(2,2,pch=23, lwd=10, col="red")  
  points(4,4,pch=23, lwd=10, col="red")  
  points(4,3,pch=23, lwd=10, col="blue")  
  points(2,1,pch=23, lwd=10, col="blue")
* 
  + Support vectors have been marked with larger points. They are at(2,2),(4,4),(4,3) and (2,1)
* 2f. The 7th point is (4,1) in the blue category. A slight movement of this point will not have an effect on the maximal margin classifer as it would not move within the support vectors of the classifier; However if it moves beyond the margin of the support vectors it will change the hyperplane
* 2g.
* plot(X1op[5:7],X2op[5:7],col="blue", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",pch=20)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", pch=20)  
  par(new=F)  
  #abline(-0.5,1)  
  abline(-0.25,1)
* 
  + The new hyper plane has the same slope as the original hyperplane but the intercept is a bit larger hence it moves this towards the red points
  + Equation of the plane is
* 2h.
* plot(X1op[5:7],X2op[5:7],col="blue", xlim=c(0,5), ylim=c(0,5),xlab="X1", ylab="X2", type="p",pch=20)  
  par(new=T)  
  plot(X1op[1:4],X2op[1:4],col="red", xlim=c(0,5), ylim=c(0,5),xlab="",ylab="",type="p", pch=20)  
  par(new=F)  
    
  points(3.5,2,col="red", pch=20)
* 

1. Q3.

* 3a.
* set.seed(1)  
  x=matrix(rnorm(20\*3\*50),ncol=50)  
  x[1:20,1]=x[1:20,1]+3  
  x[1:20,2]= x[1:20,2]+3  
  x[21:40,1]=x[21:40,1]-4  
  x[21:40,2]=x[21:40,2]-4  
  x[41:60,2]=x[41:60,2]+3  
  x[41:60,1]=x[41:60,1]-4  
  plot(x)
* 
* y=rep(NA,60)  
  y[1:20]=1  
  y[21:40]=2  
  y[41:60]=3
* 3b.
* pr.out=prcomp(x)  
   pr.out$x[,1:2]
* ## PC1 PC2  
  ## [1,] -6.0316190 0.8104626  
  ## [2,] -4.8930257 -1.8341547  
  ## [3,] -4.9721506 -0.8745783  
  ## [4,] -6.2443600 -2.9464527  
  ## [5,] -4.4891223 -2.2956106  
  ## [6,] -4.4806259 -0.3980923  
  ## [7,] -4.0600421 -3.9462011  
  ## [8,] -6.3820029 -1.3394883  
  ## [9,] -5.5263153 -2.0411504  
  ## [10,] -6.0564751 -0.3971925  
  ## [11,] -6.6470155 -3.7023009  
  ## [12,] -4.7248186 -1.7947830  
  ## [13,] -4.5981938 0.2557031  
  ## [14,] -2.7016751 -1.1502436  
  ## [15,] -4.7065350 -2.6468746  
  ## [16,] -4.9784070 -1.9088544  
  ## [17,] -4.7690020 -2.0744746  
  ## [18,] -5.6416904 -2.1008847  
  ## [19,] -5.0938766 -2.2030256  
  ## [20,] -4.7710345 -2.3769676  
  ## [21,] 5.0942207 -3.1067133  
  ## [22,] 4.6161724 -2.0894476  
  ## [23,] 4.3323709 -1.7911289  
  ## [24,] 7.8273416 -0.8952682  
  ## [25,] 4.0449609 -2.2980823  
  ## [26,] 5.0495862 -2.7300858  
  ## [27,] 4.7444407 -0.5043218  
  ## [28,] 6.6224744 -1.0225696  
  ## [29,] 5.4258099 -0.7801509  
  ## [30,] 4.2975817 -3.2855168  
  ## [31,] 4.7348472 -2.6958419  
  ## [32,] 4.4511228 -0.8810681  
  ## [33,] 4.1238047 -1.0210527  
  ## [34,] 4.5500881 -1.6091786  
  ## [35,] 5.4323430 -0.3185482  
  ## [36,] 4.9294724 -1.7263383  
  ## [37,] 6.0172545 -2.1335221  
  ## [38,] 5.8414077 -2.0752837  
  ## [39,] 4.9714181 -3.4505238  
  ## [40,] 4.6910077 -1.9655946  
  ## [41,] 0.5934246 3.2092440  
  ## [42,] 0.6486150 4.0892988  
  ## [43,] 0.4571165 2.5568799  
  ## [44,] -0.6950646 3.6618987  
  ## [45,] 1.2286169 3.5115401  
  ## [46,] -0.7644803 5.7127450  
  ## [47,] -0.8735983 2.8253423  
  ## [48,] -1.2019669 4.2231795  
  ## [49,] 0.1249432 4.1316382  
  ## [50,] -1.5107489 4.0805204  
  ## [51,] 0.2467589 2.7973085  
  ## [52,] 1.4101946 3.7683626  
  ## [53,] -0.8299349 4.7652238  
  ## [54,] 1.4080769 3.8128667  
  ## [55,] -0.9318156 1.3084279  
  ## [56,] -1.0815189 1.3320545  
  ## [57,] 0.6373622 4.7102320  
  ## [58,] 0.8897862 4.1057428  
  ## [59,] -0.7257057 3.3162644  
  ## [60,] 0.9402005 3.4266312
* plot(pr.out$x[,1:2],col=1:3, xlab="z1", ylab="z2", pch=20)  
    
   c=c(0,pr.out$rotation[1,1])  
   d=c(0,pr.out$rotation[2,1])  
   lines(5\*c,5\*d,col="purple")  
    
   c=c(0,pr.out$rotation[1,2])  
   d=c(0,pr.out$rotation[2,2])  
   lines(5\*c,5\*d,col="orange")
* 
  + The 2 principal compoents (Purple=1st principal component, Orange=2nd principal components) are plotted and shown in the graph
* 3c.
* set.seed(1)  
   km.out=kmeans(x,3,nstart=20)  
   table(y,km.out$cluster)
* ##   
  ## y 1 2 3  
  ## 1 20 0 0  
  ## 2 0 20 0  
  ## 3 0 0 20
  + All the cluster points are correctly classified
* 3d.
* set.seed(1)  
   km.out=kmeans(x,2,nstart=20)  
   table(y,km.out$cluster)
* ##   
  ## y 1 2  
  ## 1 20 0  
  ## 2 0 20  
  ## 3 20 0
  + All the points from class 3 are categorized as points from a class 1.
* 3e.
* set.seed(1)  
   km.out=kmeans(x,4,nstart=20)  
   table(y,km.out$cluster)
* ##   
  ## y 1 2 3 4  
  ## 1 0 10 0 10  
  ## 2 20 0 0 0  
  ## 3 0 0 20 0
  + Points from one of the classes are split into two clusters. Points of the remaining clusters are classified correctly
* 3f.
* set.seed(1)  
   km.out=kmeans(pr.out$x[,1:2],3,nstart=20)  
   table(y,km.out$cluster)
* ##   
  ## y 1 2 3  
  ## 1 20 0 0  
  ## 2 0 20 0  
  ## 3 0 0 20
  + All the points are classified correctly
* 3g.
* set.seed(1)  
   km.out=kmeans(scale(x),3,nstart=20)  
   table(y,km.out$cluster)
* ##   
  ## y 1 2 3  
  ## 1 7 2 11  
  ## 2 3 13 4  
  ## 3 4 10 6
  + There are more missclassifications compared to 3c; By scaling the points the distance between the points changes and hence it impacts the clustering results.

1. Q4

* 4a.
* require(tree)
* ## Loading required package: tree
* ## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
  ## logical.return = TRUE, : there is no package called 'tree'
* require(randomForest)
* ## Loading required package: randomForest  
  ## randomForest 4.6-12  
  ## Type rfNews() to see new features/changes/bug fixes.
* bodyR=load("body.RData")  
   plot(Y$Gender,Y$Weight)
* 
* set.seed(1)  
   train=sample(507,307)  
   test=-train  
   X.train=X[train,]  
   X.test=X[test,]  
   Y.test=Y[test,"Weight"]  
   Y.train=Y[train,"Weight"]  
    
   bag.body=randomForest(Y.train~.,data=X.train,mtry=21,ntree=50)  
   yhat.bag=predict(bag.body, newdata=X[-train,])  
   mean((yhat.bag-Y$Weight[-train])^2)
* ## [1] 10.76931
* rf.body=randomForest(Y$Weight~.,data=X,subset=train,mtry=7,ntree=50)  
   yhat.rf=predict(rf.body, newdata=X[-train,])  
   mean((yhat.rf-Y$Weight[-train])^2)
* ## [1] 10.21739
* plot(c(0,50),c(5,30), type="n", xlab= "# of Trees", ylab="MSE")  
   lines(rf.body$mse, col="blue", lwd=2.5)  
   lines(bag.body$mse, col="red", lwd=2.5)  
   legend(30,30,c("Bagging","Random Forest"))  
   lwd=c(2.5,2,5)  
   col=c("blue","red")  
   legend(30,30,c("Bagging","Random Forest"),lty=c(1,1),lwd=c(2.5,2.5),col=c("red","blue"))
* 
* 4b.
* varImpPlot(rf.body)
* 
* varImpPlot(bag.body)
* 
  + Top 3 variables for randomForest: Waist.Girth, Shoulder.Girth, Chest.Girth
  + Top 3 for bagging: Waist.Girth, Shoulder.Girth and Chest.Girth
  + Same variables are chosen by both methods as most important.
* 4c.
* set.seed(1)  
   rf.body=randomForest(Y$Weight~.,data=X,subset=train,mtry=7,ntree=500)  
   yhat.rf=predict(rf.body, newdata=X[-train,])  
   mean((yhat.rf-Y$Weight[-train])^2)
* ## [1] 9.904654
  + In the HW3 Solution, the PLS model had a test error of 8.65, PCR of 9.27, forward stepwise 8.63. The error here is a 9.9, that is bit higher than other methods.
* 4d.
  + The idea of using a smaller subet of 7 from 21 variables is so that we use trees from different variables and that they are uncorrelated, thus helping us to reduce the variance of the averaged trees.
  + Theorectically there are 21C7 ~116280 ways to select 7 variables from 21 variables. So theoretically adding more trees should give a better estimate.
  + Another practical way is to plot the test data error as a function of number of trees and see if the error improves.