STAT 6340 Statistical Machine Learning

Mini Project 3

Author:
Suarez Rodes, Randy

Supervisor:
Choudhary, Pankaj Ph.D.

March 22, 2021

1 Answers

Question 1.a The data is unbalanced having 1316 observations labelled as 0 (non diabetic) and 684 as 1 (diabetic). This fact can cause that classification algorithms may have a decent accuracy at the cost of a low sensitivity (setting 1 as the positive class). We explored several boxplots to investigate how our predictors affect the classes (Figure 1, Figure 2). Predictors Glucose, Pregnancies and Age separates the two classes in some degree (Figure 3, 4 and 5).

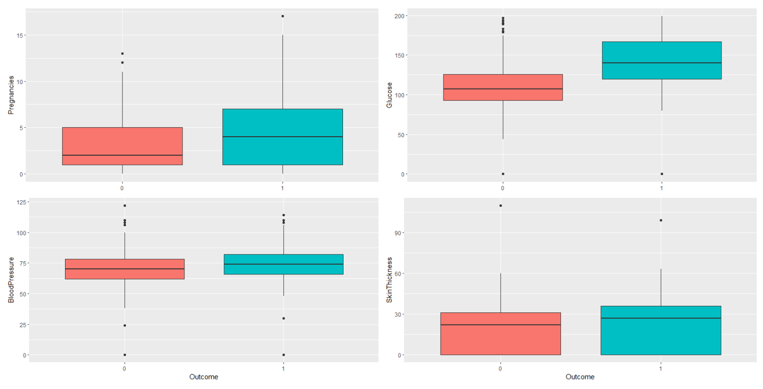


Figure 1:

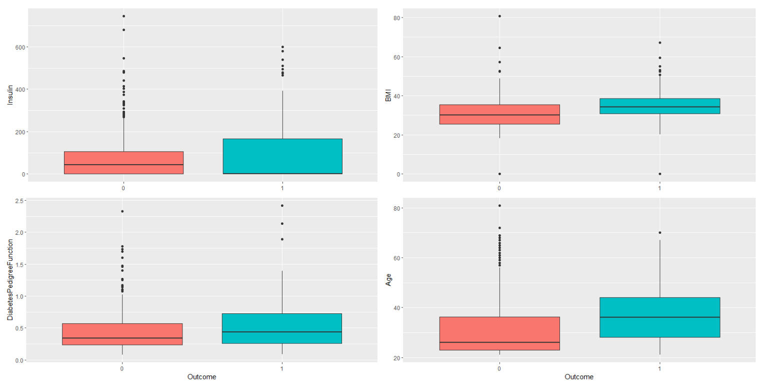


Figure 2:

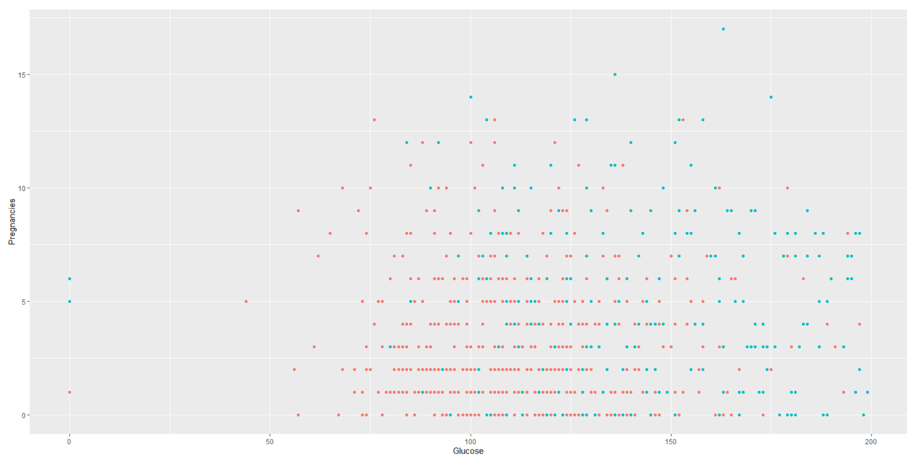


Figure 3: Pregnancies vs Glucose

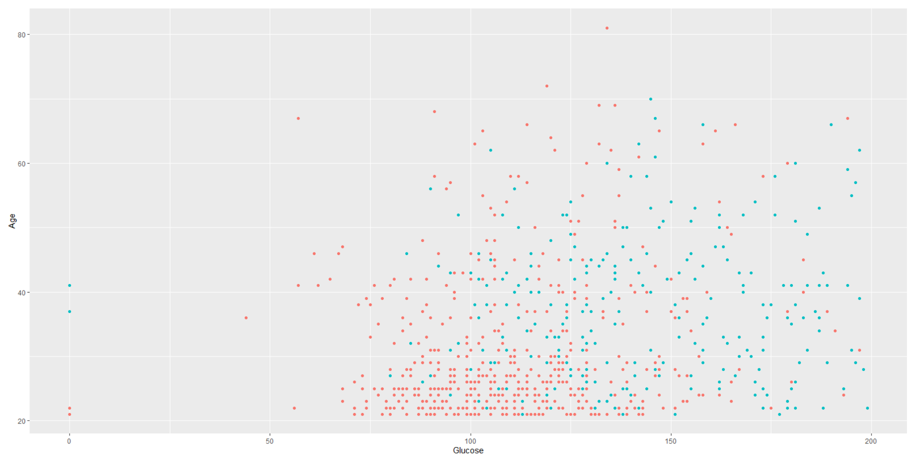


Figure 4: BMI vs Glucose

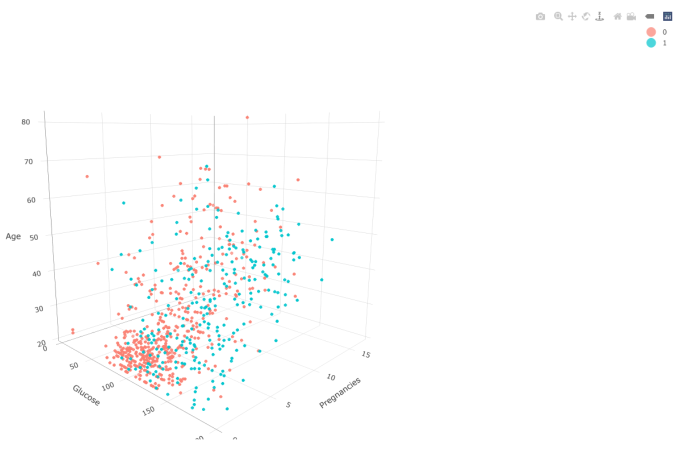


Figure 5: BMI vs Glucose vs Pregnancies

From the previous graphics we can infer a highly non-linear decision boundary might be necessary to obtain high classification accuracy.

Question 1.b To start building our model, we first fit a full model and explore possible significant predictors from the test hypothesis. We can see that the predictor SkinThickness is not relevant for our model (Table 1), therefore we proceed to drop it ending with a model where all predictors are relevant to classify Outcome (Table 2). A Chisq Test (Table 3) between the full model and the current model confirms our findings. With a p-value of 0.9024, we fail to reject the null hypothesis, this is all extra predictors are 0. We also can compare the current model with the null model. With a p-value close to 0 we reject the null hypothesis leading to the conclusion that the predictors of or our model are signifcant (Table 4).

Table 1: Full Model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-8.0265	0.4306	-18.64	0.0000
Pregnancies	0.1264	0.0200	6.32	0.0000
Glucose	0.0337	0.0022	15.15	0.0000
BloodPressure	-0.0096	0.0032	-2.97	0.0029
SkinThickness	0.0005	0.0042	0.12	0.9024
Insulin	-0.0012	0.0006	-2.15	0.0317
BMI	0.0776	0.0089	8.73	0.0000
DiabetesPedFunc	0.8878	0.1860	4.77	0.0000
Age	0.0129	0.0057	2.27	0.0232

Table 2: Resulting Model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-8.0273	0.4306	-18.64	0.0000
Pregnancies	0.1264	0.0200	6.32	0.0000
Glucose	0.0337	0.0022	15.30	0.0000
BloodPressure	-0.0096	0.0032	-2.99	0.0028
Insulin	-0.0012	0.0005	-2.32	0.0204
BMI	0.0779	0.0085	9.17	0.0000
DiabetesPedFunc	0.8895	0.1855	4.79	0.0000
Age	0.0129	0.0057	2.27	0.0234

Table 3: Analysis of Deviance vs Full Model

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
1	1992	1914.35			
2	1991	1914.33	1	0.02	0.9024

Table 4: Analysis of Deviance vs Null Model

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
1	1999	2569.41			
2	1992	1914.35	7	655.06	0.0000

Question 1.c The equation of our model is as follows:

$logit(p(x)) = x^T\beta = -8.0273 + 0.1264 * Pregnancies + 0.0337 * Glucose - 0.0096 * BloodPressure - 0.0012 * Insulin + 0.0779 * BMI + 0.8895 * DiabetesPedigreeFunction + 0.0129 * Age$

We already presented the final model on Table 2 where you can find the estimates and standard errors of each coefficient. We obtained a classification error rate of 21.6%. We can interpret odds ratio for every predictor by finding exp(coefficients). We put this result together with 95% confident intervals for the coefficients on Table 5.

We can interpret the effect of some of our predictors as follows: For a every unit increase of glucose, the odds of having diabetes (versus not having) increase approximately by a factor of 1.03, similarly for a one unit increase of BMI, the odds of having diabetes (versus not having) increase approximately by a factor of 1.08.

	Odd_Ratio	2.5 %	97.5 %
(Intercept)	0.00033	-8.88963	-7.20093
Pregnancies	1.13470	0.08745	0.16587
Glucose	1.03425	0.02944	0.03807
BloodPressure	0.99047	-0.01589	-0.00332
Insulin	0.99879	-0.00224	-0.00019
BMI	1.08099	0.06147	0.09480
DiabetesPedigreeFunction	2.43390	0.52747	1.25490
Age	1.01298	0.00171	0.02403

Table 5: Odds Ratio and Coeffs Confident Intervals

Question 2.a We fitted a model using all predictors. We obtained a classification error rate of 21.6%, a sensitivity of 56.72515% and a specificity of 89.66565%.

Question 2.b Using our own method of LOOCV we obtained an estimated test error rate of 21.95%.

Note: For questions 2.c-2.g we used the caret package to carry out LOOCV.

Question 2.c Using the caret package for LOOCV to train a logistic regression model we obtained the same estimated test error rate as in b.

Question 2.d With the final model obtained in experiment 1 we obtained an estimated test error rate of 21.85%.

Question 2.e Using LDA with a full model we obtained an estimated test error rate of 22.3%.

Question 2.f Using QDA with a full model we obtained an estimated test error rate of 24.45%.

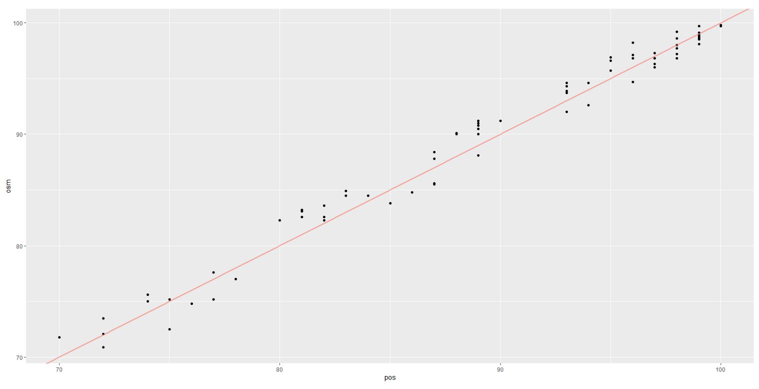
Question 2.g Using tune.knn from the e1071 package we look for an optimal K between 1-40 getting an optimal K = 1. Using KNN with this optimal K and all the predictors we obtained an estimated test error rate of 0.15%, this is only 3 out of 2000 observations where misclassified.

Question 2.h We present on Table 6 a summary of the results of LOOCV with the full model for the different classifiers. If we want a very flexible algorithm to handle the highly non-linear decision boundary necessary for our data we may decide to use the KNN classifier obtaining the best accuracy, although if we decide to trade accuracy for a less flexible model we may use logistic regression.

Classifier	Error
LR	0.2195
LDA	0.2230
QDA	0.2445
KNN	0.0015

Table 6: Results of LOOCV applied on different classifiers

Question 3.a We can appreciate a scatterplot of OSM vs POS on Figure 6. Note how several points lie close to the 45 degrees line and the majority of points follow this identity trend. The concentration of points around this line is evidence of agreement between the two methods. We can also analyze the absolute differences of the measurements by inspecting the boxplot of Figure 7. Note how 75% of the differences are below 1.7 and 90% is below or equal to 2, pointing out how close are the two methods on their results.



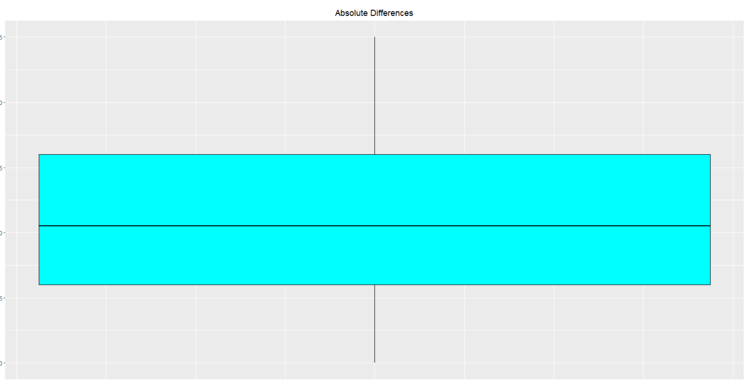


Figure 6: Scatterplot

Figure 7: Absolute Differences

Question 3.b Smaller values for θ will imply better agreement between two methods since this small value will mean that a significant high percent of data (in our case 90%) will have small differences, this is the majority of observations of the two models are considerably close when θ is relative small.

Question 3.c The natural estimate would the 90th quantile of the absolute differences, for our case $\hat{\theta} = 2$.

Question 3.d For our experiments we fixed the number of bootstrap samples to 1000. Using our own bootstrap method we obtained a bias estimate of 0.00659 (meaning that the mean of our bootstrap estimates is close to be an unbiased estimator) and we obtained a low standard error of 0.1311613. We also obtained a 95% upper confidence bound for θ equal to 2.2% saturation of hemoglobin with oxygen meaning that we are 95% certain that the TDI of the two methods is below 2.2.

Question 3.e Using the boot package we obtained similar results as in 3.d getting a bias of 0.00414 and standard error of 0.1315164. The 95% upper confidence bound for θ was also 2.2.

Question 3.f As a conclusion we can state that approximately 90% of the measurements of these two methods will agree within a difference in percent saturation of hemoglobin with oxygen of at most 2.2% with a 5% uncertainty, therefore we concluded that these two methods agree well enough to be interchanged.

2 Code

```
#Packages
library(ggplot2) #Used for graphics and visual representations,
                 #ggplots will be graph under the plot tab in rstudio
library(plotly) #Used for 3D graphics,
                #plotly graphics will be under the viewer tab in rstudio
library(ggpubr) #Used for graphics handling
library(MASS) #Used for LD and QD analysis
library(caret) #Used to handle LOOCV training
library(boot) #Used for bootstrapping

library(e1071) #Used for tuning knn

rdseed=8467 #Seed to replicate results

#Experiment 1
print("Experiment 1")

#Reading the data
diabetes = read.csv("diabetes.csv")

#Renaming predictors
names(diabetes)=c("Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
                  "Insulin",   "BMI",
                  "DiabetesPedigreeFunction", "Age","Outcome" )

#Converting Outcome to factor
diabetes$Outcome=as.factor(diabetes$Outcome)

####Question 1.a####

#Exploring data
print(table(diabetes$Outcome))
#We can notice how the data is unbalanced for our classes

#Visualizing Boxplots of several predictors
g_1=ggplot(diabetes,aes(Outcome,Pregnancies,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
g_2=ggplot(diabetes,aes(Outcome,Glucose,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
g_3=ggplot(diabetes,aes(Outcome,BloodPressure,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
g_4=ggplot(diabetes,aes(Outcome,SkinThickness,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
print(ggarrange(g_1+theme(axis.title.x = element_blank()),
                g_2+theme(axis.title.x = element_blank()),
                g_3, g_4, ncol = 2, nrow = 2))

g_5=ggplot(diabetes,aes(Outcome,Insulin,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
g_6=ggplot(diabetes,aes(Outcome,BMI,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
g_7=ggplot(diabetes,aes(Outcome,DiabetesPedigreeFunction,fill=Outcome))+
  geom_boxplot()+  theme(legend.position = "none")
g_8=ggplot(diabetes,aes(Outcome,Age,fill=Outcome))+geom_boxplot()+
  theme(legend.position = "none")
print(ggarrange(g_5+theme(axis.title.x = element_blank()),
                g_6+theme(axis.title.x = element_blank()),
                g_7, g_8, ncol = 2, nrow = 2))
```

```
#From the previous boxplots we consider Glucose, Pregnancies and Age are good
#candidates predictors for our model
print(ggplot(diabetes,aes(x=Glucose,y=Pregnancies,color=Outcome))+geom_point()+
        theme(legend.position = "none"))

print(ggplot(diabetes,aes(x=Glucose,y=Age,color=Outcome))+geom_point()+
        theme(legend.position = "none"))

#3D graph
plotly3 = plot_ly(diabetes, x = ~Glucose, y = ~Pregnancies, z = ~Age,
              color = ~Outcome, colors = c('Salmon', 'Turquoise3'))%>%
  add_markers(size=0.5) %>% layout(scene = list(xaxis = list(title = 'Glucose'),
                                   yaxis = list(title = 'Pregnancies'),
                                   zaxis = list(title = 'Age')))
print(plotly3)

#Storing true classes
actual=diabetes$Outcome

####Question 1.b####

#Fitting the full model
m_Full=glm(Outcome~.,data=diabetes,family = binomial)
print(summary(m_Full))

#Null model
m0=glm(Outcome~1,data=diabetes,family = binomial)

#Dropping one predictor at a time
m1=glm(Outcome~.-SkinThickness,data=diabetes,family=binomial) #dropping SkinThickness
print(summary(m1))

#Analysis of Deviance with the full model
print(anova(m1, m_Full, test = "Chisq"))

#Analysis of Deviance with the null model
print(anova(m0, m1, test = "Chisq"))

####Question 1.c####

#Summary of final model
summary(m1)

#Equation logit(p(x))=x^t*beta
#x^t*beta=-8.0273146 + 0.1263707*Pregnancies + 0.0336810*Glucose -0.0095806*BloodPressure
#          -0.0012123*Insulin + 0.0778743*BMI + 0.8894946*DiabetesPedigreeFunction + 0.0128944*Age

#95% confidence interval of coefficients
ci95=confint(m1)

#Taking exponent of coefficients for interpretation purposes
odd_ratio=exp(m1$coefficients)

#Putting all together
print(cbind(Odd_Ratio=odd_ratio,ci95))

#Predicted Classes
class.predict=ifelse(m1$fitted.values >= 0.5, 1, 0)

#Classification error rate
print(mean(class.predict!=actual))
```

```
#Experiment 2
print("Experiment 2")

####Question 2.a####

m_Full=glm(Outcome~.,data=diabetes,family = binomial)
lr.prob = m_Full$fitted.values
lr.class = ifelse(lr.prob >= 0.5, 1, 0)

#Confusion Matrix
CM=table(lr.class, actual,dnn=c("predicted","actual"))
print(CM)

#Finding accuracy
acc_train=sum(diag(CM))/sum(CM)
#Finding error
print(paste("Misclassifcation Error Rate Experiment 1 model:",1-acc_train))

#Finding sensitivity
print(paste("Sensitivity:",CM[2,2]/sum(CM[,2])))

#Finding especificity
print(paste("Especificity",CM[1,1]/sum(CM[,1])))

####Question 2.b####

n=nrow(diabetes) #number of observations

#Applying LOOCV on a full model for logistic regression
class_LOOCV=sapply(1:n,function(i){

  m_i=glm(Outcome~.,data=diabetes[-i,],family = binomial)

  lr_i_class=as.factor(ifelse(predict(m_i,diabetes[i,],
                                      type = "response") >= 0.5, 1, 0))

  (lr_i_class != diabetes$Outcome[i])

})

#Estimated test error rate
print(mean(class_LOOCV))

####Question 2.c####

#Error Dataframe to track errors of each classifier
error.df=data.frame(Error=rep(0,4))
row.names(error.df)=c("LR","LDA","QDA","KNN")


#Defining the training control for the train method of caret
control=trainControl(method = "LOOCV",seeds = rep(rdseed,n+1), number = 1)


#LOOCV on full model of logistic regression
mloocv = train(
  form = Outcome ~ .,
  data = diabetes,
  trControl = control,
  method = "glm",
  family = "binomial"
)
```

```
#Estimated test error rate, subtracting accuracy
mloocv_error=(1-mloocv$results[1,2])
print(mloocv_error)
error.df["LR","Error"]=mloocv_error

####Question 2.d####

#LOOCV on logistic regression model from experiment 1
m1_loocv = train(
  form = Outcome ~ .-SkinThickness,
  data = diabetes,
  trControl = control,
  method = "glm",
  family = "binomial"
)

#Estimated test error rate, subtracting accuracy
m1error_loocv=(1-m1_loocv$results[1,2])
print(m1error_loocv)


####Question 2.e####

#LOOCV on full model of LDA
lda_loocv = train(
  form = Outcome ~ .,
  data = diabetes,
  trControl = control,
  method = "lda"
)

#Estimated test error rate, subtracting accuracy
lda_error=(1-lda_loocv$results[1,2])
print(lda_error)
error.df["LDA","Error"]=lda_error

####Question 2.f####

#LOOCV on full model of QDA
qda_loocv = train(
  form = Outcome ~ .,
  data = diabetes,
  trControl = control,
  method = "qda"
)

#Estimated test error rate, subtracting accuracy
qda_error=(1-qda_loocv$results[1,2])
print(qda_error)
error.df["QDA","Error"]=qda_error

####Question 2.g####

#Finding optimal k between 5-40 neighbors
set.seed(rdseed)
knntunned=tune.knn(diabetes[,-9], diabetes$Outcome,k = 1:40,
                   tunecontrol = tune.control(cross=n))

opt_k=knntunned$best.parameters[[1]] #optimal K = 1
```

```
knn_loocv = train(
  form = Outcome ~ .,
  data=diabetes,
  trControl = control,
  method = "knn",
  tuneGrid=data.frame(k=opt_k)
  )

#Estimated test error rate, subtracting accuracy
knn_loocv_error=(1-knn_loocv$results[[1,2]])
print(knn_loocv_error)
error.df["KNN","Error"]=knn_loocv_error


####Question 2.h####
print(error.df)#comparison of all classification techniques


#Experiment 3
print("Experiment 3")

oxygen_saturation = read.delim("oxygen_saturation.txt")

####Question 3.a####

#Scatterplot OSM vs POS
gox=ggplot(oxygen_saturation,aes(x=pos,y=osm))+geom_point(color='black')+
  geom_abline(slope=1,intercept = 0,color='salmon',size=1,alpha=0.7)

print(gox)

D=oxygen_saturation[,1]-oxygen_saturation[,2]
abs_D=abs(D) #Absolute Differences

#Boxplot of Absolute Differences
gbox=ggplot()+geom_boxplot(fill='cyan', color="black",aes(y=abs_D))+
  ggtitle("Absolute Differences")+theme(plot.title=element_text(hjust=0.5))+
  theme(axis.title.y=element_blank(),axis.title.x=element_blank(),
      axis.text.x=element_blank(),
      axis.ticks.x=element_blank())

print(gbox)

####Question 3.b####

#Answered on report.

####Question 3.c####

#Natural estimate is the 90% quantile
theta_hat=quantile(abs_D,0.9)[[1]]
print(theta_hat)

####Question 3.d####

#Number of bootstrap samples
nb=1000

#Generating Bootstrap Samples
set.seed(rdseed)
boot_sample=replicate(nb, sample(abs_D, replace=TRUE),simplify = FALSE)

#Finding Bootstrap Estimates
```

```
boot_estimates=sapply(boot_sample, function(x){quantile(x,0.9)[[1]]})

#bias estimate
print(paste("Bias estimate:",mean(boot_estimates)-theta_hat))

#std error
print(paste("Std error estimate:",sd(boot_estimates)))

#95% upper confidence bound
print(paste("95% upper bound:",sort(boot_estimates)[ceiling(.95*nb)]))


####Question 3.e####

#Auxiliary function to be used with the boot package
quantile.fn=function(x,indices)
{
  quantile(x[indices],0.9)[[1]]
}

#Generating bootstrap estimates
set.seed(rdseed)
theta.boot=boot(abs_D,quantile.fn,nb)
print(theta.boot)

#95% upper confidence bound
print(paste("95% upper bound:",sort(theta.boot$t)[ceiling(.95*nb)]))
```