Assignment 1

Biomedical Data Science (MATH11174), 22/23, Semester 2

March 9, 2023

Due on Thursday, 9th of March 2023, 5:00pm

> ! Pay Attention
>
> The assignment is marked out of 100 points, and will contribute to *20%* of your final mark. The aim of this assignment is to produce a precise report in biomedical studies with the help of statistics and machine learning. Please complete this assignment using **Quarto/Rmarkdown file and render/knit this document only in PDF format** and submit using the **gradescope link on Learn**. You can simply click render on the top left of Rstudio (`Ctrl+Shift+K`). If you cannot render/knit to PDF directly, open **Terminal** in your RStudio (`Alt+Shift+R`) and type `quarto tools install tinytex`, otherwise please follow this link. If you have any code that does not run you will not be able to render nor knit the document so comment it as you might still get some grades for partial code.
> **Clear and reusable code will be rewarded**. Codes without proper indentation, choice of variable identifiers, **comments**, error checking, etc will be penalised. An initial code chunk is provided after each subquestion but **create as many chunks as you feel is necessary** to make a clear report. Add plain text explanations in between the chunks when required to make it easier to follow your code and reasoning. Ensure that all answers containing multiple values should be presented and formatted with `kable()` and `kable_styling()` or using Markdown syntax. All plots must be displayed with clear title, label and legend.

Problem 1 (25 points)

Files `longegfr1.csv` and `longegfr2.csv` (available on Assessment > Assignment 1) contain information regarding a longitudinal dataset containing records on 250 patients. For each

subject, eGFR (**estimated glomerular filtration rate, a measure of kidney function**) was collected at irregularly spaced time points: variable `fu.years` contains the follow-up time (that is, the distance from baseline to the date when each eGFR measurement was taken, expressed in years).

Problem 1.a (4 points)

- Convert the files to data table format and merge in an appropriate way into a single data table.
- Order the observations according to subject identifier and follow-up time.
- Print first 10 values of the new dataset using `head()`.

```
# s2319494
# First let us read the csv files using the fread() function.
lgfr1<-fread('longegfr1.csv')
lgfr2<-fread('longegfr2.csv')
# Now we convert the data in lgfr1, lgfr2 into data tables using
# the setDT() function.
lgfr1<-setDT(lgfr1)
lgfr2<-setDT(lgfr2)
# The next step is to merge the data tables. In order to do so, we need
# common columns or keys. On observing, id and fu.years are the keys. The
# merge.data.table() function is then used to merge the data tables lgfr1
# and lgfr2.
lgfr<-merge.data.table(lgfr1,lgfr2,by.x = c('id','fu.years'),
                       by.y = c('ID','fu.years'),all = TRUE)

# The next step is to order the entire table by id and follow-up time. This
# can be done conveniently using with() and thus specifying the columns to
# order by using order(). The results are stored in lgfr12
lgfr12<-lgfr[with(lgfr,order(id,fu.years))]
# View first 10 rows using head().
head(lgfr12,10)
```

```
    id fu.years sex baseline.age  egfr
 1:  1   0.0000   0         65.5 76.48
 2:  1   0.1533   0         65.5 47.36
 3:  1   0.6899   0         65.5 94.87
 4:  1   1.1882   0         65.5 52.12
 5:  1   1.8398   0         65.5 91.91
 6:  1   2.2806   0         65.5 76.52
```

```
 7:  1   3.3895   0         65.5 46.79
 8:  1   3.7563   0         65.5 35.56
 9:  1   4.5229   0         65.5 28.41
10:  1   5.3607   0         65.5 20.85
```

Problem 1.b (6 points)

- Compute the average eGFR and length of follow-up for each patient.
- Print first 10 values of the new dataset using `head()`.
- Tabulate the number of patients with average eGFR in the following ranges: $(0, 15]$, $(15, 30]$, $(30, 60]$, $(60, 90]$, $(90, \texttt{max(eGFR)})$.
- Count and report the number of patients with missing average eGFR.

```
# Our objective now is to compute average or mean of eGFR and total length
# of follow up time (fu.years) by patient.
# We do so by computing two new columns avg_eGFR and sum(fu.years) in the process.
# And on specifying by='id', R does the computation by patient. The result is
# stored in Avg_eGFR_fu.
Avg_eGFR_fu<-lgfr12[,.(avg_eGFR=mean(egfr),length_of_followup=sum(fu.years)),by='id']

# Next we print first 10 values using head()
head(Avg_eGFR_fu,10)
```

```
    id  avg_eGFR length_of_followup
 1:  1  43.04333           53.8341
 2:  2  38.93294           15.7618
 3:  3  85.72000           34.7487
 4:  4  76.59308           31.8055
 5:  5        NA          192.1423
 6:  6  85.66435           89.7441
 7:  7  64.21758          131.7151
 8:  8  66.28333            9.1226
 9:  9  86.35750           26.5627
10: 10 107.00429           20.4599
```

```
# Now let us specify the given range labels that will be used later on.
ranges<-c('(0,15]','(15,30]','(30,60]','(60,90]','(90,max(egfr))')

# Now let us count and store the number of patients in each of those ranges.
num_patients<-c(nrow(Avg_eGFR_fu[avg_eGFR>0 & avg_eGFR<=15]),
```

```
                nrow(Avg_eGFR_fu[avg_eGFR>15 & avg_eGFR<=30]),
                nrow(Avg_eGFR_fu[avg_eGFR>30 & avg_eGFR<=60]),
                nrow(Avg_eGFR_fu[avg_eGFR>60 & avg_eGFR<=90]),
                nrow(Avg_eGFR_fu[avg_eGFR>90]))

# Now let us form a data table to better visualize the number of rows or
# samples in each range using the 2 vectors we computed earlier.
egfr_ranges<-data.table(ranges,num_patients)
egfr_ranges
```

```
          ranges num_patients
1:         (0,15]            1
2:        (15,30]            9
3:        (30,60]           83
4:        (60,90]           82
5: (90,max(egfr))           36
```

```
# Now let us print the number of patients with missing eGFR using cat().
# This can simply be done by counting the number of rows with avg_eGFR as
# NA using nrow() and using is.na() to find the NA rows.
cat('The number of patients with missing average eGFR is',
    nrow(Avg_eGFR_fu[is.na(avg_eGFR)]))
```

```
The number of patients with missing average eGFR is 39
```

Problem 1.c (6 points)

- For patients with average eGFR in the (90, max(eGFR)) range, collect their identifier, sex, age at baseline, average eGFR, time of last eGFR reading and number of eGFR measurements taken in a data table.
- Print the summary of the new dataset.

```
# The next objective is to collect specific data of the given patients.
# For convenience, let us add the average eGFR that we computed earlier
# for Avg_eGFR_fu to the original data table lgfr121.
lgfr121<-lgfr12[,`:=` (avg_eGFR=mean(egfr)),
                by='id']

```

```
# Now we use lgfr121 to return a set of specific details of the patients
# whose ids are greater than 90. For a given id,sex,baseline age and average
# eGFR are all constant. So we simply use the unique() function to return
# a single value for each.

# The number of measurments by id can simply be returned using .N operator.
# Finally, the last eGFR reading by id can be returned using .SD and
# specifying index of row as .N. This retrieves the last row by id which
# had required fu.years value.
lgfr122<-lgfr121[avg_eGFR>90,.(sex=unique(sex),
                               baseline_age=unique(baseline.age),
                               avg_eGFR=unique(avg_eGFR),
                               last_reading_time=.SD[.N,fu.years],
                               Nmeasurments=.N),by='id']

# Let us print the summary of the new data table lgfr122 using summary().
summary(lgfr122)
```

```
      id              sex          baseline_age      avg_eGFR
Min.   : 10.00   Min.   :0.0000   Min.   :22.10   Min.   : 90.04
1st Qu.: 79.75   1st Qu.:0.0000   1st Qu.:48.05   1st Qu.: 94.49
Median :137.00   Median :0.0000   Median :56.45   Median :105.61
Mean   :139.19   Mean   :0.3889   Mean   :57.54   Mean   :105.65
3rd Qu.:215.75   3rd Qu.:1.0000   3rd Qu.:67.83   3rd Qu.:113.91
Max.   :250.00   Max.   :1.0000   Max.   :90.90   Max.   :147.69
last_reading_time  Nmeasurments
Min.   :0.000     Min.   : 1.00
1st Qu.:1.275     1st Qu.: 4.00
Median :3.469     Median : 7.00
Mean   :3.253     Mean   :11.11
3rd Qu.:5.201     3rd Qu.:10.00
Max.   :6.590     Max.   :57.00
```

```
# Finally, we print the first 10 rows of the data table using head().
head(lgfr122,10)
```

```
   id sex baseline_age  avg_eGFR last_reading_time Nmeasurments
1: 10   0         50.4 107.00429            5.1964            7
2: 14   0         65.1 116.09200            4.0986           10
3: 25   0         40.1  95.35625            4.2847            8
```

```
 4: 31   0        74.8 113.59250          1.4675          8
 5: 33   0        74.2 116.35000          1.6016          4
 6: 45   1        24.9  91.25000          0.0000          1
 7: 49   1        68.2 128.25800          6.1602          5
 8: 52   1        56.3  93.31544          6.4805         57
 9: 79   0        65.6  91.45057          5.2156         35
10: 80   0        67.7 106.09600          2.2834          5
```

Problem 1.d (9 points)

For patients 3, 37, 162 and 223:

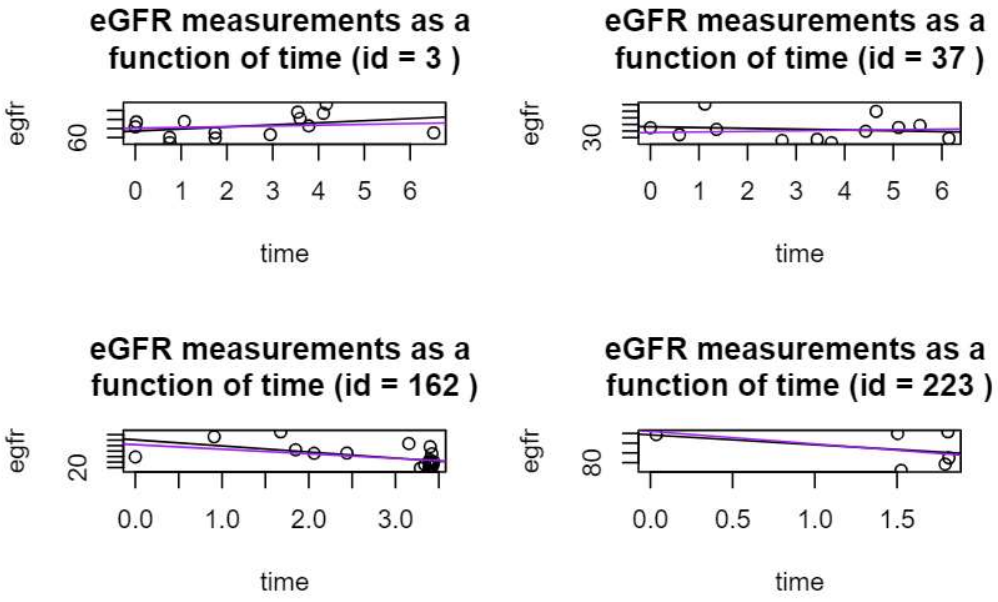
- Plot the patient's eGFR measurements as a function of time.
- Fit a linear regression model and add the regression line to the plot.
- Report the 95% confidence interval for the regression coefficients of the fitted model.
- Using a different colour, plot a second regression line computed after removing the extreme eGFR values (one each of the highest and the lowest value).

(All plots should be displayed in the same figure. The plots should be appropriately labelled and the results should be accompanied by some explanation as you would communicate it to a colleague with a medical background with a very little statistical knowledge.)

```r
# Next we will be plotting eGFR v time. We will also be plotting two regression lines
# for the data points for the given ids. The first with all data points and the
# second with removing the extreme values. The 1st line is in black while
# the 2nd is in purple.

# First let's partition the screen into 4 parts using par().
par(mfrow=c(2,2))
# The vector of ids to plot.
a<-c(3,37,162,223)

for(i in a){
  # Next for each id:
  # we plot egfr as a function of time.
  p1<-plot(lgfr12[id==i]$fu.years,lgfr12[id==i]$egfr,
           xlab = 'time',ylab = 'egfr',
           main = paste('eGFR measurements as a \nfunction of time (id =',i,')'))
  # Now we model the first linear regression line for egfr and fu.years using lm().
  regr1<-lm(egfr ~ fu.years,data = lgfr12[id==i])
  # We then pass the regression object regr1 to abline() that draws the
```

```
    # regression line. This line is added to the previous scatterplot.
    p1<-p1+abline(regr1)
    # Now we report 95% confidence value using confint() function.
    cat('\n95% confidence interval (id =',i,')\n',
        confint(regr1,'fu.years',level = 0.95),'\n')
    # We now find out the extreme values i.e, the maximum and minimum
    # values using max() and min() respectively. To exclude NA values
    # from the computation, we set na.rm=TRUE.
    max_egfr<-max(lgfr12[id==i]$egfr,na.rm = TRUE)
    min_egfr<-min(lgfr12[id==i]$egfr,na.rm = TRUE)
    # Now filter rows by id
    new_lgfr3<-lgfr12[id==i]
    # Next exclude the extreme values from the filtered rows.
    new_lgfr3<-new_lgfr3[egfr!=max_egfr & egfr!=min_egfr]
    # Now create a second linear regression model with
    # the new values.
    regr11<-lm(egfr ~ fu.years,data = new_lgfr3)
    # The resultant regression line is added to the final plot.
    p1<-p1+abline(regr11,col='purple')
    # Print resultant plot (Repeat process for each id)
    p1
  }
```

```
95% confidence interval (id = 3 )
 -3.151128 12.25612


95% confidence interval (id = 37 )
 -3.595705 2.37859


95% confidence interval (id = 162 )
 -9.257727 -1.872262
```




```
95% confidence interval (id = 223 )
 -85.93757 45.9659
```

```
### Explanation of the plots:
# The regression plots for all 4 ids can be visualized as shown below.
# The black line indicates the first regression line on all data points
# by given id. The purple line indicates the second regression line.
# This line was formed out of the same data points as before except we
# exclude extreme values. The ids are 3,37,162 and 223 respectively.
# Also observe how the slopes of the new regression lines (in purple)
# changed after we removed extreme values (or outliers).

# On removing these values, one can expect that the correlation between
# the variables would increase, be it much more positive (tending to 1)
# or much more negative (tending to -1).

# We also observe a change in slope of the line. As the model does not
# try to fit-in those extreme values anymore, we observe a increase or
# decrease in slope accordingly.This can be observed in the plots below.
```

Problem 2 (25 points)

The MDRD4 and CKD-EPI equations are two different ways of estimating the glomerular filtration rate (eGFR) in adults:

$$\texttt{MDRD4} = 175 \times (\texttt{SCR})^{-1.154} \times \texttt{AGE}^{-0.203}[\times 0.742 \text{ if female}][\times 1.212 \text{ if black}]$$

, and

$$\texttt{CKD-EPI} = 141 \times \min(\texttt{SCR}/\kappa, 1)^{\alpha} \times \max(\texttt{SCR}/\kappa, 1)^{-1.209} \times 0.993^{\texttt{AGE}}[\times 1.018 \text{ if female}][\times 1.159 \text{ if black}]$$

, where:

- SCR is serum creatinine (in mg/dL)
- κ is 0.7 for females and 0.9 for males
- α is -0.329 for females and -0.411 for males

Problem 2.a (7 points)

For the `scr.csv` dataset,

- Examine a summary of the distribution of serum creatinine and report the inter-quartile range.
- If you suspect that some serum creatinine values may have been reported in µmol/L convert them to mg/dL by dividing by 88.42.
- Justify your choice of values to convert and examine the distribution of serum creatinine following any changes you have made.

```
# First lets read the required file using fread().
scr_d<-fread('scr.csv')
# Then convert it to a data table using setDT().
scr_d<-setDT(scr_d)
# Now order the data table by scr values using the with() function.
scr_d<-scr_d[with(scr_d,order(scr))]
# Next, print a summary of the serum creatinine values:
summary(scr_d$scr)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.400   0.900   1.300   3.072   2.800  76.000      18
```

```r
# Delete NA values using na.omit()
scr_d<-na.omit(scr_d)
# Now lets store the first and third quartile values in q.
# This will be useful later on.
q<-quantile(scr_d$scr,c(0.25,0.75))
# The inter-quartile range is computed using the IQR() function.
iqr<-IQR(scr_d$scr)
# Print the IQR using cat().
cat('The Inter-Quartile Range is',iqr)
```

```
The Inter-Quartile Range is 1.9
```

```r
# Now the upper and lower bound for the whiskers can be computed as follows:
# LB = Q1 - 1.5*IQR and UB = Q3 + 1.5*IQR
lb<-q[1]-1.5*iqr
ub<-q[2]+1.5*iqr
# Now the assumption is that the values above ub or below lb are outliers.
# In this case, they are the values to be scaled to mg/dL.
scr_d_t<-scr_d[,
      scr:=ifelse(scr>ub,
                  scr/88.42,
                  scr)]
### Choice of Values:
# There are some scr values that may have been reported in umol/litre.
# One easy way to  find out which ones is to simply find the "outliers"
# in a boxplot. The data is first sorted in ascending order and then compute
# the lower bound and upper bound using the following formulae:
# LB = Q1 - 1.5*IQR and UB = Q3 + 1.5*IQR
# From here on, pretty much any value lesser than LB or greater than UB
# is assumed to be an outlier and thus the values that are in umol/litre
# and are thus converted to mg/dL by dividing by 88.42.
```

Problem 2.b (11 points)

- Compute the eGFR according to the two equations using the newly converted `SCR` values.
- Report (rounded to the second decimal place) mean and standard deviation of the two eGFR vectors and their Pearson correlation coefficient.
- Report the same quantities according to strata of MDRD4 eGFR: $(0-60)$, $(60-90)$ and (>90).

- Print first 15 values for both datasets using `head()`.

```
# The two functions below compute the new scr values according to the 2
# equations specified earlier.
# MDRD4_calculate() calculates new scr values by equation 1.
MDRD4_calculate<-function(scr,sex,ethnic,age){
  e_factor<-ifelse(ethnic=='Black',1.212,1)
  f_factor<-ifelse(sex=='Female',0.742,1)
  eq<-175*scr^(-1.154)*age^(-0.203)*f_factor*e_factor
  return(eq)
}
# Call the function on our data table scr_d_t and pass required parameters.
mdrd4<-MDRD4_calculate(scr_d_t$scr,scr_d_t$sex,scr_d_t$ethnic,scr_d_t$age)
# Now report the mean and standard deviation of the new scr values.
# While the mean is the 4th value in the summary, the Standard deviation is
# computed using sd(). The results are rounded to 2 decimal places using round().
cat('Mean of mdrd4 scr values:',round(summary(mdrd4)[4],2),'\n')
```

```
Mean of mdrd4 scr values: 188.98
```

```
cat('Standard Deviation of mdrd4 scr values:',round(sd(mdrd4),2),'\n\n') #na
```

```
Standard Deviation of mdrd4 scr values: 359.03
```

```
# CKD_EPI_calculate() calculates new scr values by equation 2.
CKD_EPI_calculate<-function(scr,sex,ethnic,age){

  l<-length(scr)
  K<-ifelse(sex=='Female',0.7,0.9)
  A<-ifelse(sex=='Female',-0.329,-0.411)
  e_factor<-ifelse(ethnic=='Black',1.159,1)
  f_factor<-ifelse(sex=='Female',1.018,1)

  min_factor<-max_factor<-rep(0,l)

  for(i in 1:l){
  min_factor[i]<-min(scr[i]/K[i],1)
  max_factor[i]<-max(scr[i]/K[i],1)
  }
```

```
  eq<-141*min_factor^(A)*max_factor^(-1.209)*0.993^(age)*f_factor*e_factor
  return(eq)
}
# Call the function on our data table scr_d_t and pass required parameters.
ckd_epi<-CKD_EPI_calculate(scr_d_t$scr,scr_d_t$sex,scr_d_t$ethnic,scr_d_t$age)
# Report mean and standard deviation like before and also the Pearson
# Correlation coefficient using cor() function.
cat('Mean of ckd epi scr values:',round(summary(ckd_epi)[4],2),'\n')
```

```
Mean of ckd epi scr values: 85.84
```

```
cat('Standard Deviation of ckd epi scr values:',round(sd(ckd_epi),2),'\n')
```

```
Standard Deviation of ckd epi scr values: 64.27
```

```
cat('Pearson correlation coefficient:',round(cor(mdrd4,ckd_epi),2),'\n\n')
```

```
Pearson correlation coefficient: 0.86
```

```
# Form a new data table new_scr_values using the values computed earlier.
new_scr_values<-data.table('MDRD4'=mdrd4,'CKD-EPI'=ckd_epi)
# Now we filter the values by several strata of MDRD4.
s1<-new_scr_values[MDRD4>0 & MDRD4<60]
s2<-new_scr_values[MDRD4>60 & MDRD4<90]
s3<-new_scr_values[MDRD4>90]

# Now we print the mean, standard deviation and correlation coefficient for each strata.
cat('MDRD4 (0-60 Strata):\nMDRD4 Equation Mean:'
    ,round(summary(s1$MDRD4)[4],2),
'\nCKD-EPI Equation Mean:'
,round(summary(s1$`CKD-EPI`)[4],2),
'\nPearson Coefficient:'
,round(cor(s1$MDRD4,s1$`CKD-EPI`),2),'\n\n')
```

```
MDRD4 (0-60 Strata):
MDRD4 Equation Mean: 31.9
CKD-EPI Equation Mean: 33.15
Pearson Coefficient: 0.99
```

```
cat('MDRD4 (60-90 Strata):\nMDRD4 Equation Mean:'
    ,round(summary(s2$MDRD4)[4],2),
'\nCKD-EPI Equation Mean:'
,round(summary(s2$`CKD-EPI`)[4],2),
'\nPearson Coefficient:'
,round(cor(s2$MDRD4,s2$`CKD-EPI`),2),'\n\n')
```

```
MDRD4 (60-90 Strata):
MDRD4 Equation Mean: 73.41
CKD-EPI Equation Mean: 80.18
Pearson Coefficient: 0.93
```

```
cat('MDRD4 (>90 Strata):\nMean:'
    ,round(summary(s3$MDRD4)[4],2),
'\nCKD-EPI Equation Mean:'
,round(summary(s3$`CKD-EPI`)[4],2),
'\nPearson Coefficient:'
,round(cor(s3$MDRD4,s3$`CKD-EPI`),2),'\n\n')
```

```
MDRD4 (>90 Strata):
Mean: 466.01
CKD-EPI Equation Mean: 156.02
Pearson Coefficient: 0.95
```

```
# Print first 15 values of new data table.
head(new_scr_values,15)
```

```
      MDRD4   CKD-EPI
 1: 184.9777 137.81686
 2: 167.9425 115.32906
 3: 169.6141 117.78528
 4: 135.3041 119.37447
 5: 195.2410 145.41646
```

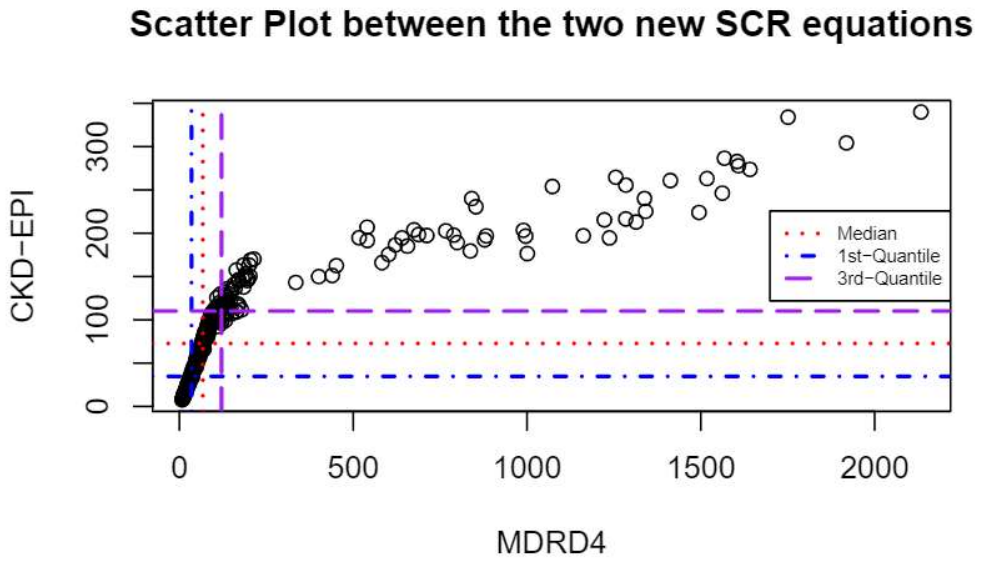
```
 6: 144.8688 129.87348
 7: 120.9417  96.01479
 8: 132.2497 115.25445
 9: 170.1717 145.32828
10: 147.8353 113.65115
11: 169.6141 117.78528
12: 150.3311 134.51609
13: 202.6025 150.61469
14: 134.0324 117.70908
15: 138.0812 122.77628
```

Problem 2.c (7 points)

- Produce a scatter plot of the two eGFR vectors, and add vertical and horizontal lines (i.e.) corresponding to median, first and third quantiles.
- Is the relationship between the two eGFR equations linear? Justify your answer.

```r
# Let's first retrieve the median, 1st and 3rd quartile from the summary
# of the 2 new scr values
# (They are in the 3rd, 2nd and 5th indicies respectively):
mdrd4_v<-summary(new_scr_values$MDRD4)[c(3,2,5)]
ckd_v<-summary(new_scr_values$`CKD-EPI`)[c(3,2,5)]
# Next, a scatter plot is plotted between the 2 set of values. The title,
# xlabel, ylabel can be specified using additional parameters as follows:
plot(x=new_scr_values$MDRD4,y=new_scr_values$`CKD-EPI`,
     xlab = 'MDRD4',ylab='CKD-EPI',
     main = 'Scatter Plot between the two new SCR equations')
# Now we add three lines.
# These indicate median, 1st and 3rd quartile respectively.
# We do the same for MDRD4 and then for CKD-EPI values that give us 6
# lines overall (3 Vertical and 3 Horizontal).
# The width of each line and type of line can be customized
# using lty and lwd.
abline(v=mdrd4_v,col=c('red','blue','purple'),
       lty=c(3,4,5),lwd=c(2,2,2))
abline(h=ckd_v,col=c('red','blue','purple'),
       lty=c(3,4,5),lwd=c(2,2,2))
# Then a legend is added using legend() to indicate the values
# each line points to.
legend('right',legend=c('Median','1st-Quantile','3rd-Quantile'),
       col=c('red','blue','purple'),
```

```
25          lty=c(3,4,5),lwd=c(2,2,2),cex=0.6)
```




```
### The relationship between the equations:
# The relationship between the two eGFR equations appears to follow a
# linear trend until we go past the third quantile. From there,
# we clearly oberve that the relationship is not linear anymore.
# We see the points scattered with increase in MDRD4 and overall
# it appears as if it is forming a curve thus hinting at a quadratic
# relationship.
```

Problem 3 (31 points)

You have been provided with electronic health record data from a study cohort. Three CSV (Comma Separated Variable) files are provided on learn.

The first file is a cohort description file `cohort.csv` file with fields:

- `id` = study identifier
- `yob` = year of birth
- `age` = age at measurement
- `bp` = systolic blood pressure
- `albumin` = last known albuminuric status (categorical)
- `diabetes` = diabetes status

The second file `lab1.csv` is provided by a laboratory after measuring various biochemistry levels in the cohort blood samples. Notice that a separate lab identifier is used to anonymise results from the cohort. The year of birth is also provided as a check that the year of birth aligns between the two merged sets.

- `LABID` = lab identifier
- `yob` = year of birth
- `urea` = blood urea
- `creatinine` = serum creatinine
- `glucose` = random blood glucose

To link the two data files together, a third linker file `linker.csv` is provided. The linker file includes a `LABID` identifier and the corresponding cohort `id` for each person in the cohort.

Problem 3.a (6 points)

- Using all three files provided on learn, load and merge to create a single data table based dataset `cohort.dt`. This will be used in your analysis.
- Perform assertion checks to ensure that all identifiers in `cohort.csv` have been accounted for in the final table and that any validation fields are consistent between sets.
- After the checks are complete, drop the identifier that originated from `lab1.csv` dataset `LABID`.
- Ensure that a single `yob` field remains and rename it to `yob`.
- Ensure that the `albumin` field is converted to a factor and the ordering of the factor is `1="normo"`, `2="micro"`, `3="macro"`.
- Print first 10 values of the new dataset using `head()`.

```
# Read the 3 datsets.
d1<-fread('cohort.csv')
d2<-fread('lab1.csv')
d3<-fread('linker.csv')
# Convert datasets to data tables.
d1<-setDT(d1)
d2<-setDT(d2)
d3<-setDT(d3)
# Next, we merge the data tables by key column.
# First d1 and d3 are merged by common key column 'id'.
d4<-merge.data.table(d1,d3,by.x = 'id',by.y = 'id',all = TRUE)
# Next we merge the resultant data table d4 with d2
# by c('LABID','yob') to create final data table cohort.dt.
cohort.dt<-merge.data.table(d4,d2,
                            by.x = c('LABID','yob'),
                            by.y = c('LABID','yob'),all = TRUE)
```

```
# Next we perform a simple assertion check to check whether all ids
# in d1 were accounted for in the final data table.
# The idea is that if the intersection of d1 and cohort.dt ids gives
# number of ids in d1, that simply means that no ids were lost in
# merge process.
length(intersect(d1$id,cohort.dt$id))==length(d1$id) # returns TRUE
```

```
[1] TRUE
```

```
# Next we delete a row called LABID by initializing it to NULL.
cohort.dt[,LABID:=NULL]

# The categorical variable albumin is converted to numeric by following
# normo=1,micro=2,macro=3 convention.
cohort.dt$albumin<-c(normo=1,micro=2,macro=3)[cohort.dt$albumin]
# Print first 10 rows.
head(cohort.dt,10)
```

```
    yob      id age bp diabetes albumin urea creatinine glucose
1: 1986 PID_285  33 80        0       1 37.0    106.104     100
2: 1980 PID_153  39 70        1       1 20.0     70.736     121
3: 1951  PID_13  68 70        1       2 72.0    185.682     208
```

```
 4: 1965 PID_110  54 70        1      NA 50.1    167.998     233
 5: 1953 PID_222  66 70        1       2 30.0    150.314     248
 6: 2002 PID_103  17 60        0       1 32.0    185.682      92
 7: 1954 PID_200  65 80        0       1 37.0    132.630      92
 8: 1955 PID_378  64 70        0       1 27.0     61.894      97
 9: 1964 PID_267  55 80        0       1 17.0    106.104     133
10: 1996 PID_271  23 80        0       1 34.0     97.262     111
```

Problem 3.b (10 points)

- Create a copy of the dataset where you will impute all missing values.
- Update any missing age fields using the year of birth.
- Perform mean imputation for all other continuous variables by writing a single function called `impute.to.mean()` and impute to mean, impute any categorical variable to the mode.
- Print first 15 values of the new dataset using `head()`.
- Compare each distribution of the imputed and non-imputed variables and decide which ones to keep for further analysis. Justify your answer.

```r
# Now let us create a copy of the dataset using cohort.dt %>% copy().
# We also fill missing values by calculating age using corresponding
# yob as follows:
cohort.dt.impute<-cohort.dt %>% copy() %>%
  .[, age:=ifelse(is.na(age), 2019-yob,age)]
# A function is created which fills the missing values of a
# column with the mean.
impute.to.mean <- function(x) {
              # find which values are missing
              na.idx <- is.na(x)
              # replace NAs with the mean computed over observed values
              x[na.idx] <- mean(x, na.rm = TRUE)
              # return the vector with imputed mean values
              return(x)
}
# A function to calculate mode of a column.
mode_exclude_na <- function(x) {
  uq <- na.omit(unique(x))
  t <- tabulate(match(x, uq))
  uq[t == max(t)]
}
# A function is created which fills the missing values
```

```r
# of a column with the mode.
impute.to.mode <- function(x) {
              # find which values are missing
              na.idx <- is.na(x)
              # replace NAs with the mean computed over observed values
              x[na.idx] <- mode_exclude_na(x)
              # return the vector with imputed values
              return(x)
}
# Impute miising values in bp, urea, creatinine and glucose
# columns with mean
cohort.dt.impute$bp <- impute.to.mean(cohort.dt.impute$bp)
cohort.dt.impute$urea <- impute.to.mean(cohort.dt.impute$urea)
cohort.dt.impute$creatinine<- impute.to.mean(cohort.dt.impute$creatinine)
cohort.dt.impute$glucose<- impute.to.mean(cohort.dt.impute$glucose)
# Impute miising values in albumin and diabetes columns with mode
cohort.dt.impute$albumin<- impute.to.mode(cohort.dt.impute$albumin)
cohort.dt.impute$diabetes<- impute.to.mode(cohort.dt.impute$diabetes)
# Return first 15 values.
head(cohort.dt.impute,15)
```

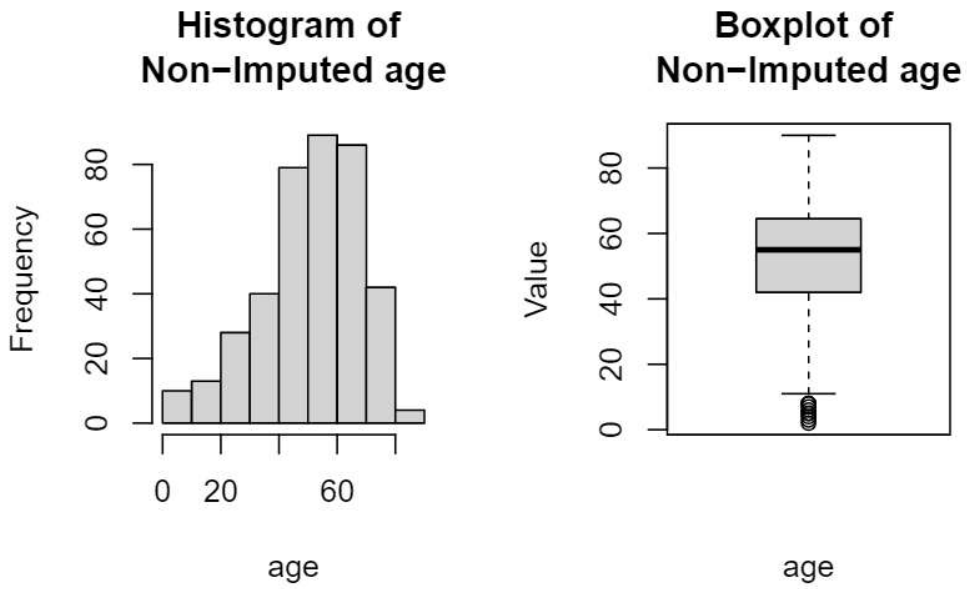
	yob	id	age	bp	diabetes	albumin	urea	creatinine	glucose
1:	1986	PID_285	33	80	0	1	37.00000	106.1040	100.0000
2:	1980	PID_153	39	70	1	1	20.00000	70.7360	121.0000
3:	1951	PID_13	68	70	1	2	72.00000	185.6820	208.0000
4:	1965	PID_110	54	70	1	1	50.10000	167.9980	233.0000
5:	1953	PID_222	66	70	1	2	30.00000	150.3140	248.0000
6:	2002	PID_103	17	60	0	1	32.00000	185.6820	92.0000
7:	1954	PID_200	65	80	0	1	37.00000	132.6300	92.0000
8:	1955	PID_378	64	70	0	1	27.00000	61.8940	97.0000
9:	1964	PID_267	55	80	0	1	17.00000	106.1040	133.0000
10:	1996	PID_271	23	80	0	1	34.00000	97.2620	111.0000
11:	1964	PID_105	55	90	1	1	88.00000	176.8400	143.0000
12:	1940	PID_375	79	80	0	1	44.00000	106.1040	111.0000
13:	1961	PID_89	58	110	0	3	52.00000	194.5240	251.0000
14:	1998	PID_24	21	70	0	1	57.42572	271.6664	148.0365
15:	1981	PID_349	38	80	0	1	19.00000	44.2100	99.0000

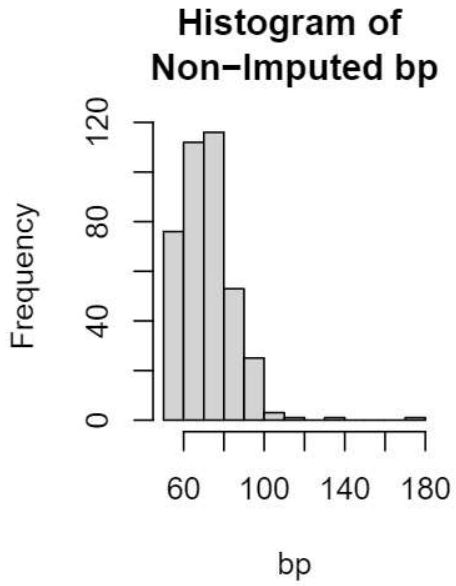
```r
#Answer in this chunk
```

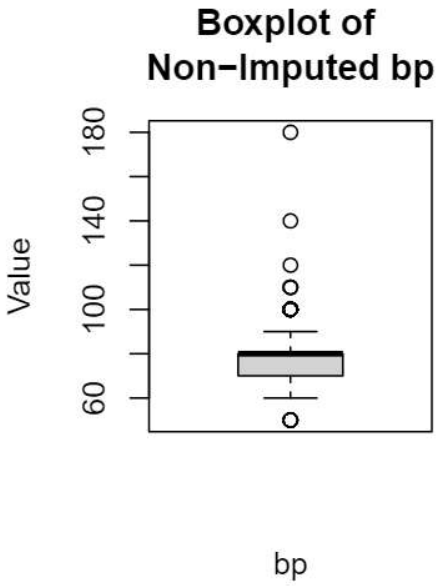
```r
# Store the imputed values in a data table dx2.
dx2<-data.table(age=cohort.dt.impute$age,
                bp=cohort.dt.impute$bp,
                urea=cohort.dt.impute$urea,
                creatinine =cohort.dt.impute$creatinine,
                albumin=cohort.dt.impute$albumin,
                glucose=cohort.dt.impute$glucose,
                diabetes=cohort.dt.impute$diabetes)
# Store the corresponding initial or Non-imputed values in a data table dx1.
dx1<-data.table(age=cohort.dt$age,
                bp=cohort.dt$bp,
                urea=cohort.dt$urea,
                creatinine =cohort.dt$creatinine,
                albumin=cohort.dt$albumin,
                glucose=cohort.dt$glucose,
                diabetes=cohort.dt.impute$diabetes)

# Function to plot distributions of imputed variables.
sbsplots_i <- function(varname, vars){
  # Partition screen into 2 parts
  par(mfrow = c(1,2))
  # Plot histogram using hist()
  hist(vars[,varname],
     main = paste0("Histogram of \nImputed ", varname),
     xlab = varname)
  # Plot boxplot using boxplot()
  boxplot(vars[,varname],
        main = paste0("Boxplot of \nImputed ", varname),
        xlab = varname,
        ylab = "Value")
}
# Function to plot distributions of imputed variables.
sbsplots_ni <- function(varname, vars){
  # Partition screen into 2 parts
  par(mfrow = c(1,2))
  # Plot histogram using hist()
  hist(vars[,varname],
     main = paste0("Histogram of \nNon-Imputed ", varname),
     xlab = varname)
  # Plot boxplot using boxplot()
  boxplot(vars[,varname],
```

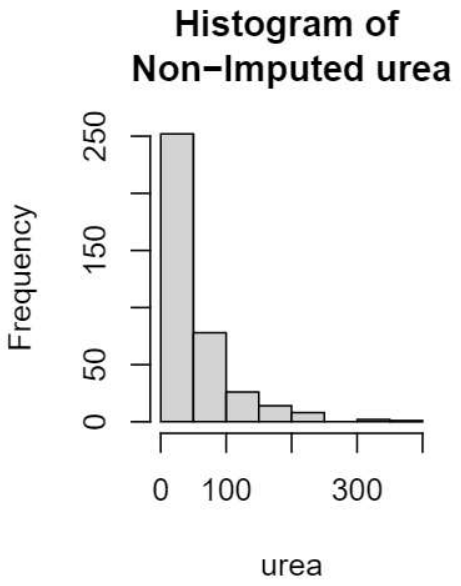
```
        main = paste0("Boxplot of \nNon-Imputed ", varname),
        xlab = varname,
        ylab = "Value")
}
```

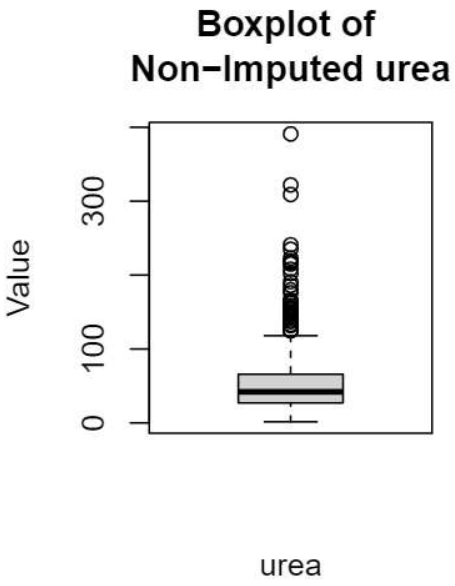
```
# Store required Non-imputed column names
numcols1 <- dx1 %>% colnames
# Use sapply() to apply sbsplots_ni() to required columns.
sapply(numcols1, sbsplots_ni, vars = data.frame(dx1)) # Non-Imputed
```

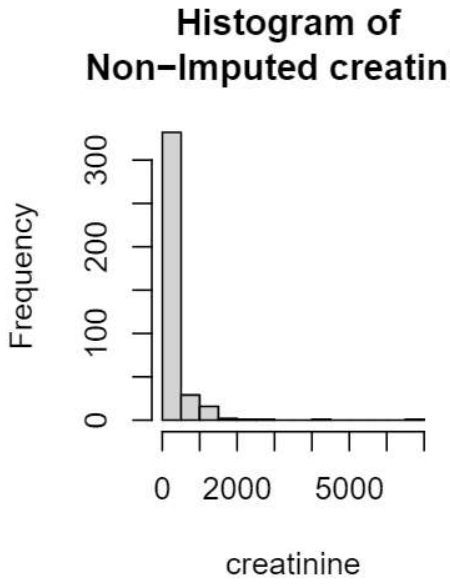


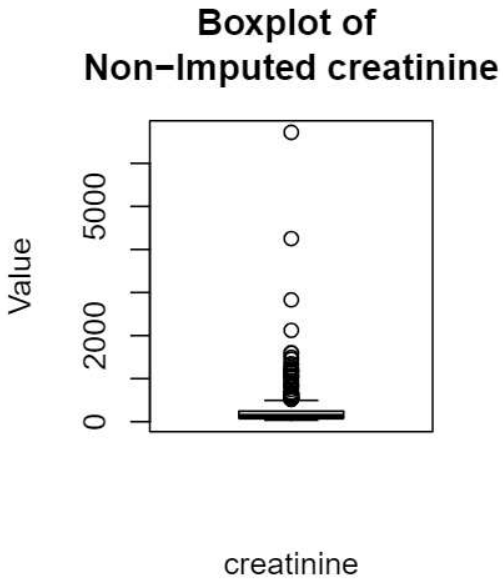

Histogram of
Non−Imputed bp
Frequency
0
40
80
120
60
100
140
180
bp

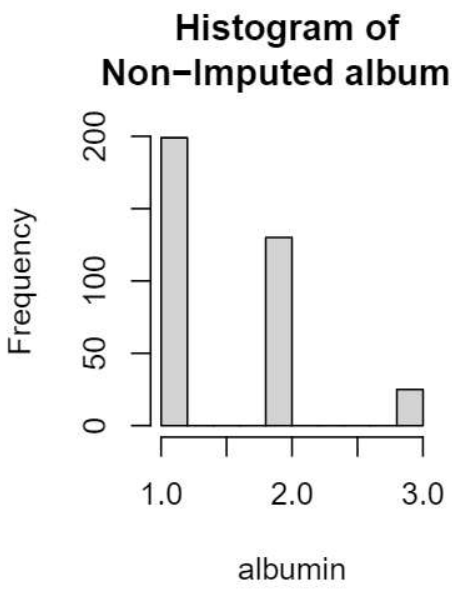

Boxplot of
Non−Imputed bp
Value
60
100
140
180
bp

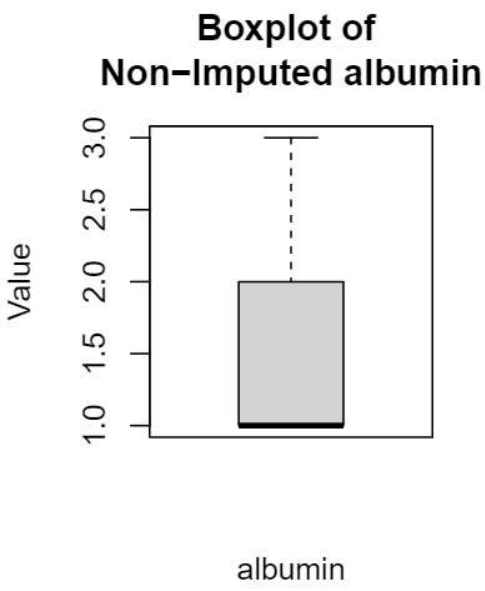

Histogram of
Non−Imputed urea
Frequency
0
50
150
250
0
100
300
urea

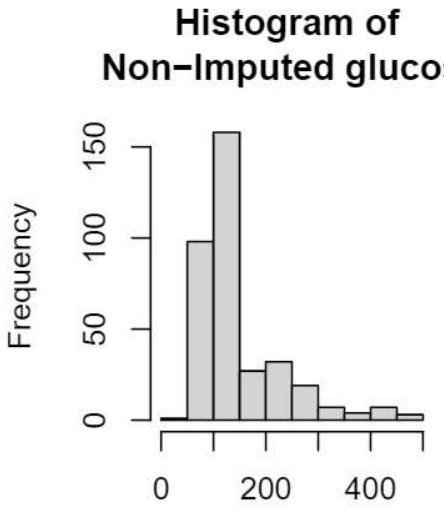

Boxplot of
Non−Imputed urea
Value
0
100
300
urea

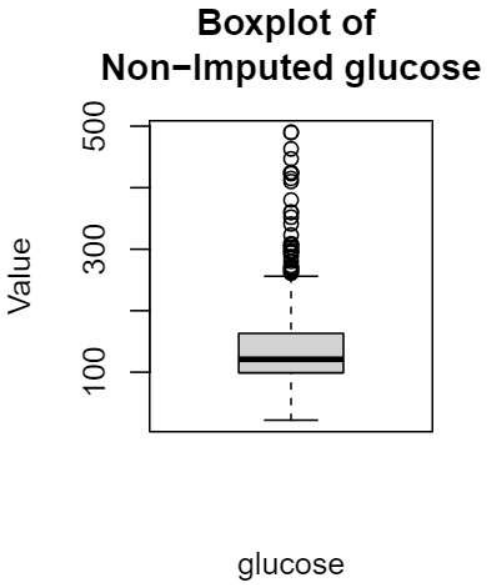

Histogram of
Non−Imputed creatinine
Frequency
0
100
200
300
0
2000
5000
creatinine

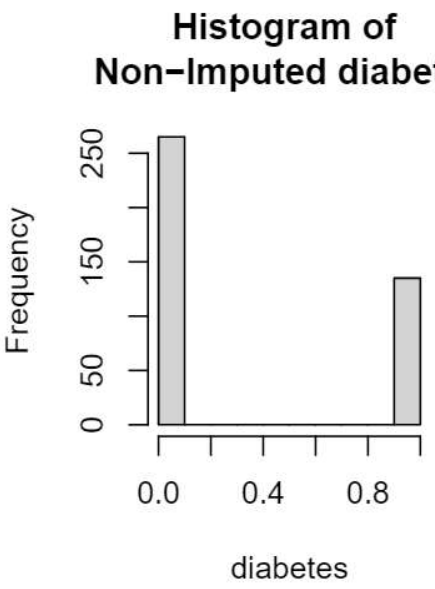

Boxplot of
Non−Imputed creatinine
Value
0
2000
5000
creatinine

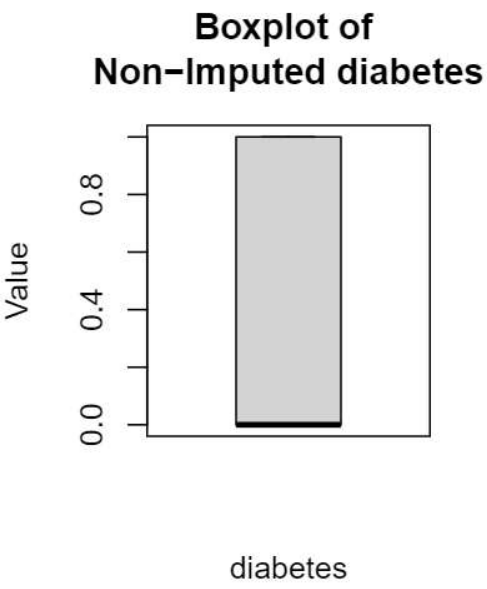

Histogram of
Non−Imputed albumin
Frequency
0
50
100
200
1.0
2.0
3.0
albumin


Boxplot of
Non−Imputed albumin
Value
1.0
1.5
2.0
2.5
3.0
albumin





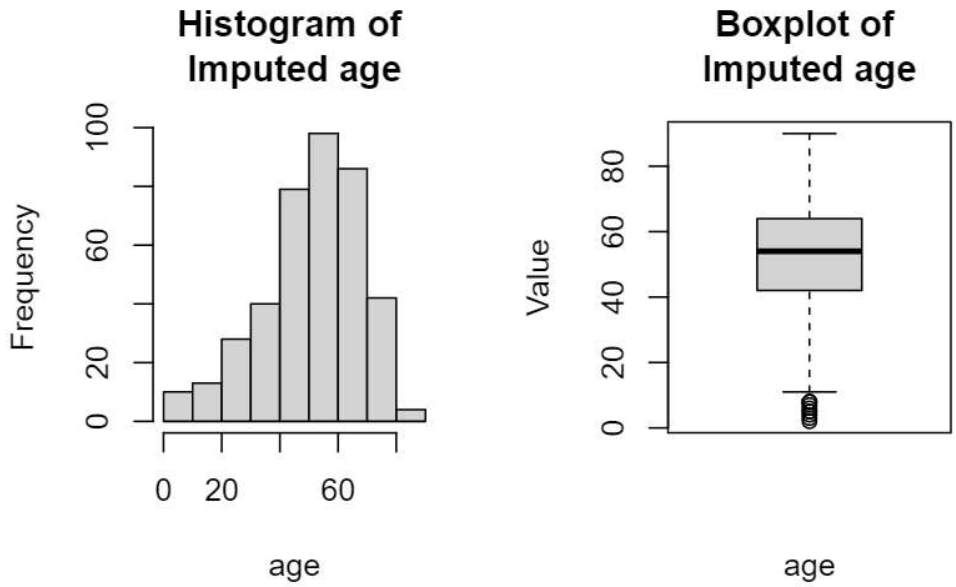


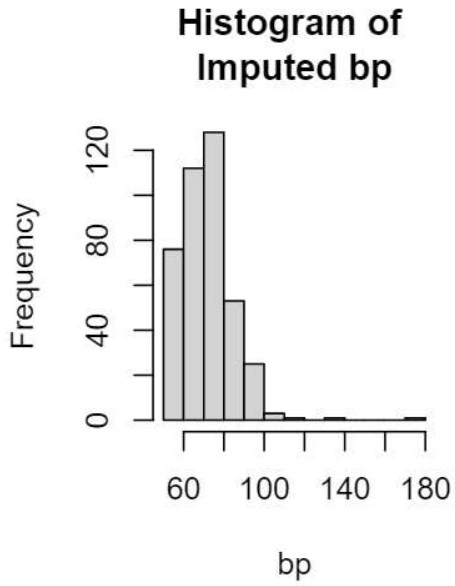


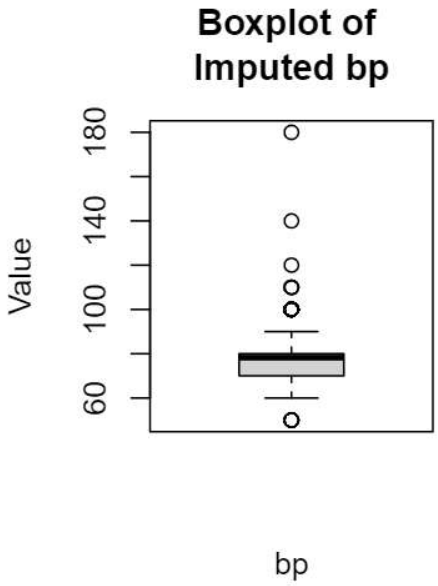
	age	bp	urea	creatinine	albumin	glucose
stats	numeric,5	numeric,5	numeric,5	numeric,5	numeric,5	numeric,5
n	391	388	381	383	354	356
conf	numeric,2	numeric,2	numeric,2	numeric,2	numeric,2	numeric,2
out	numeric,10	numeric,36	numeric,38	numeric,51	numeric,0	numeric,34
group	numeric,10	numeric,36	numeric,38	numeric,51	numeric,0	numeric,34
names	""	""	""	""	""	""

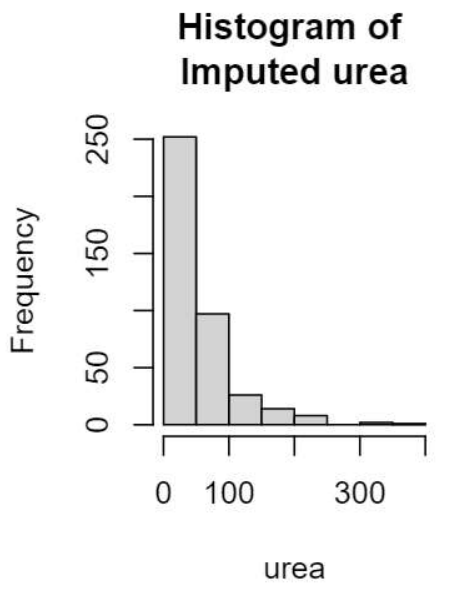
```
      diabetes
stats numeric,5
n     400
conf  numeric,2
out   numeric,0
group numeric,0
names ""
```

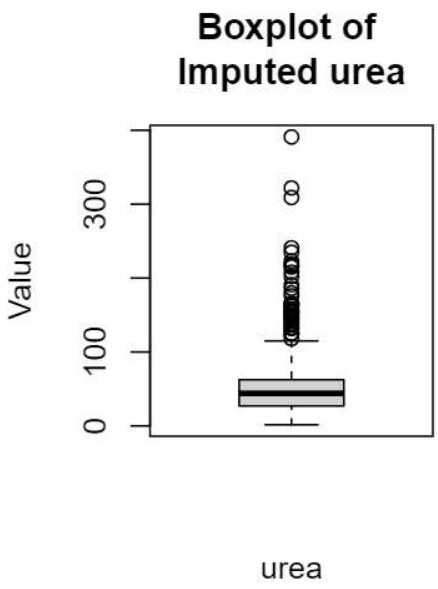
```
# Store required imputed column names
numcols2 <- dx2 %>% colnames
# Use sapply() to apply sbsplots_i() to required columns.
sapply(numcols2, sbsplots_i, vars = data.frame(dx2)) # Imputed
```

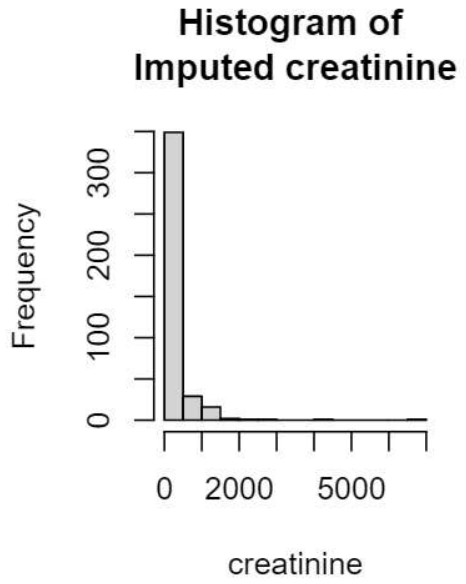


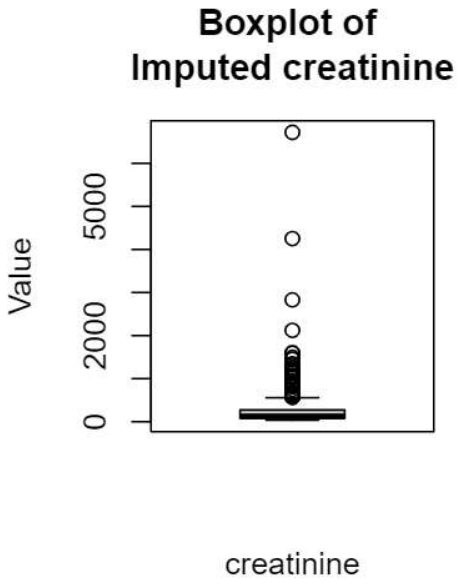

Histogram of
Imputed bp
Frequency
0
40
80
120
60
100
140
180
bp

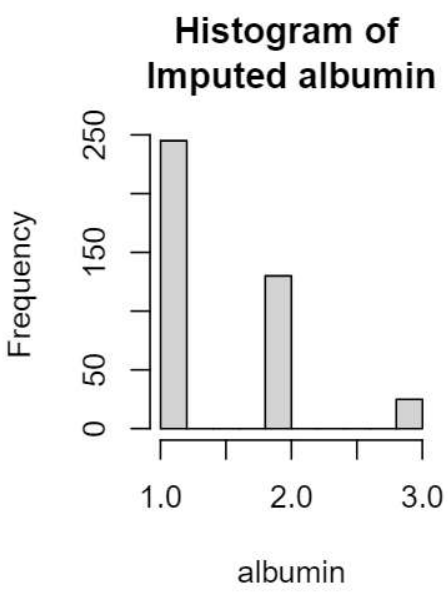

Boxplot of
Imputed bp
Value
60
100
140
180
bp

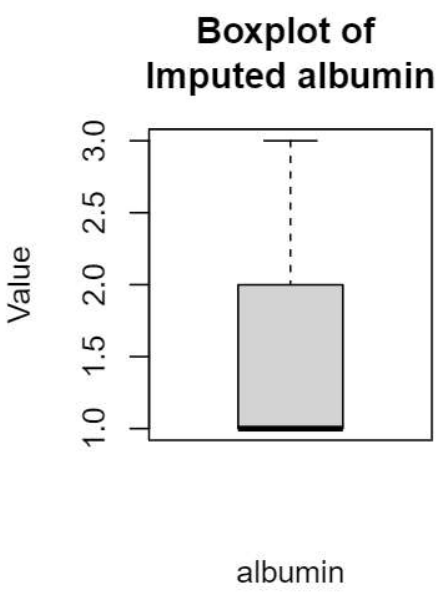

Histogram of
Imputed urea
Frequency
0
50
150
250
0
100
300
urea

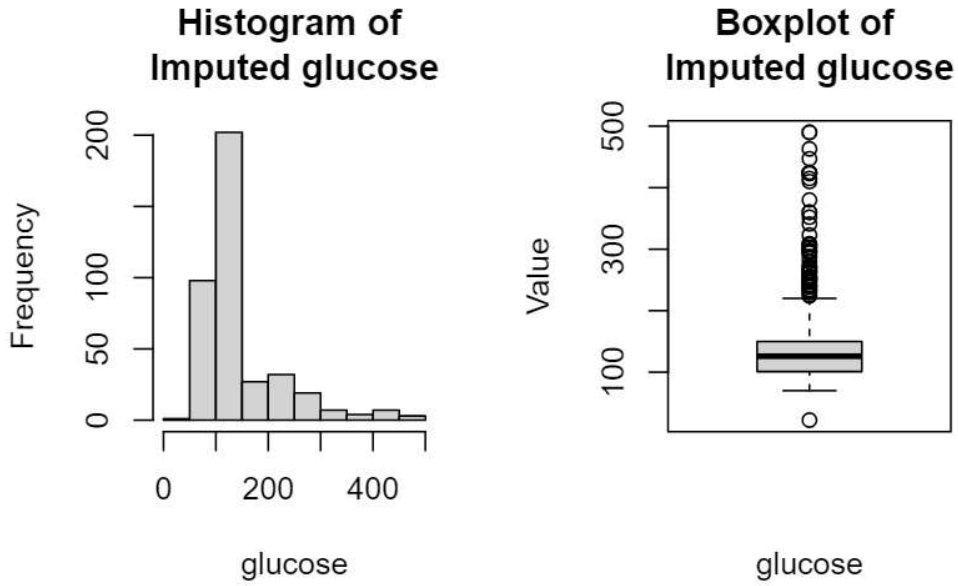

Boxplot of
Imputed urea
Value
0
100
300
urea

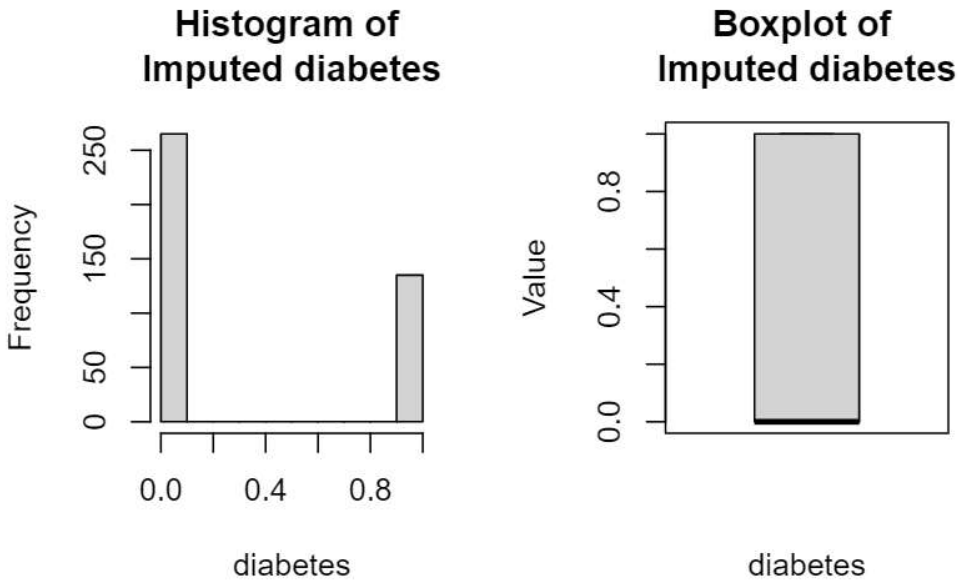

Histogram of
Imputed creatinine
Frequency
0
100
200
300
0
2000
5000
creatinine


Boxplot of
Imputed creatinine
Value
0
2000
5000
creatinine


Histogram of
Imputed albumin
Frequency
0
50
150
250
1.0
2.0
3.0
albumin


Boxplot of
Imputed albumin
Value
1.0
1.5
2.0
2.5
3.0
albumin





	age	bp	urea	creatinine	albumin	glucose
stats	numeric,5	numeric,5	numeric,5	numeric,5	numeric,5	numeric,5
n	400	400	400	400	400	400
conf	numeric,2	numeric,2	numeric,2	numeric,2	numeric,2	numeric,2
out	numeric,10	numeric,36	numeric,39	numeric,44	numeric,0	numeric,53
group	numeric,10	numeric,36	numeric,39	numeric,44	numeric,0	numeric,53
names	""	""	""	""	""	""

```
      diabetes
stats numeric,5
n     400
conf  numeric,2
out   numeric,0
group numeric,0
names ""
```

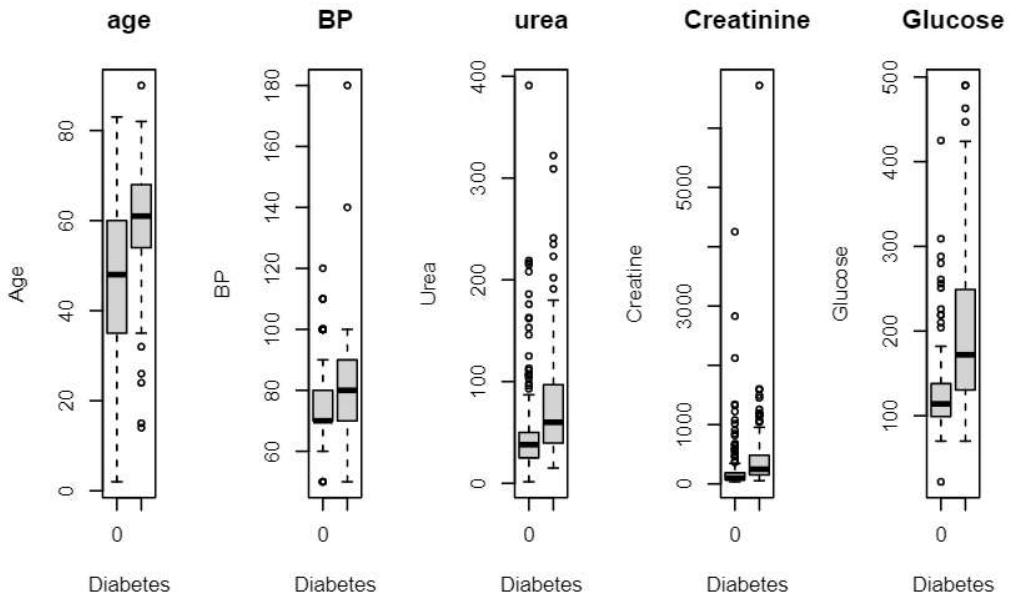
```
### Explanation:
# On looking at the boxplot for given imputed and non-imputed variables,
# the imputed variables appear to have a more symmetric distribution of
# values. This can be observed with Age for instance. After imputation
# with mean, the median of the Age values is much closer to the center
# whereas in case of the Non-Imputed Age, the median is much closer
# to the second quartile making it skewed to the right. This can be
# observed in case of blood pressure as well.

# In case of urea, the Non-imputed values were more skewed towards the
# left and on imputation, the Median has moved much closer to the centre.

# We might not see much of a difference incase of creatinine but we see a
# small reduction in outliers on imputation. Finally for the discrete
# variable albumin, there is no difference in the boxplot although
# the histogram shows an increase in "normo" values on imputation to mode.
# The variable 'Diabetes' does not see any change overall as there are
# a negligible amount of null values pre-imputation.
```

Problem 3.c (6 points)

- Plot a single figure containing boxplots of potential predictors for `diabetes` grouped by cases and controls. (Hint : `par(mfrow=c(1,5))`)
- Use these to decide which predictors to keep for future analysis.
- For any categorical variables create a table instead. Justify your answers.

```
# Split page for 5 plots:
par(mfrow=c(1,5))
# Now to plot box plots of potential predictor varibales
# (age,bp,urea,creatinine,glucose)
  boxplot(dx2$age~dx2$diabetes,
          main='age',xlab = 'Diabetes',ylab = 'Age')
```

```r
boxplot(dx2$bp~dx2$diabetes,
        main='BP',xlab = 'Diabetes',ylab = 'BP')
boxplot(dx2$urea~dx2$diabetes,
        main='urea',xlab = 'Diabetes',ylab = 'Urea')
boxplot(dx2$creatinine~dx2$diabetes,
        main='Creatinine',xlab = 'Diabetes',ylab = 'Creatine')
boxplot(dx2$glucose~dx2$diabetes,
        main='Glucose',xlab = 'Diabetes',ylab = 'Glucose')
```




```r
# Frequency table: albumin v diabetes.
# Rows indicate Albumin values and columns indicate diabetes values.
table(dx2$albumin,dx2$diabetes)
```

```
   0   1
1 192  53
2  61  69
3  12  13
```

```r
### Justification:
# On observing the boxplots of several predictor variables in relation
# to diabetes, age and glucose appear to be more viable. Not only do they
```

```r
# have the least number of outliers but the data points are also
# more symmetrically disturbuted compared to other variables.
# For instance, there are too many outliers in creatinine and bp
# (diabetes =0) samples are skewed to the right.
```

Problem 3.d (9 points)

- Use your findings from the previous exercise and fit an appropriate model of `diabetes` with two predictors.
- Print a summary and explain the results as you would communicate it to a colleague with a medical background with a very little statistical knowledge.

```r
# Based on previous findings, we plot a logistic regression model with
# choosen predictor variables using glm().
diabetes_model1<-glm(diabetes~age+glucose,data = dx2,family = 'binomial')
# Let's view a summary of the model.
summary(diabetes_model1)
```

```
Call:
glm(formula = diabetes ~ age + glucose, family = "binomial",
    data = dx2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7764  -0.7237  -0.4284   0.6580   2.7543

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.066415   0.661022  -9.177  < 2e-16 ***
age          0.049688   0.009481   5.241 1.60e-07 ***
glucose      0.018031   0.002532   7.120 1.08e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 511.49  on 399  degrees of freedom
Residual deviance: 368.02  on 397  degrees of freedom
AIC: 374.02
```

```
Number of Fisher Scoring iterations: 5
```

```
### Summary Explanation:
# The first line indicates the logistic regression model with the parameters
# that were inputted.

# The second line contains "deviance residuals" that are simply a 5-number
# summary of the residuals (just like in box-plot). We have minimum and
# maximum values that are roughly the same distance from origin if you think
# about it. We also see quantiles 1 followed by median and quantile 3 at last.

# Third line has the coefficients! The intercepts are values when the
# features are at 0. The estimates give expected change in response due
# to a unit change in the feature. Next we have standard errors that are
# simply errors involved in calculating estimates that allow to construct
# confidence intervals for that feature.

# Fourth line has the Z test values that describe how far the estimated value
# value is from null hypothesis value.

# Next we have the "p-values" (p(>|z|)). This is the probability of observing
# a value as extreme as z. We reject the null hypothesis if this value is
# very low. These probabilities follow Significance Codes below. The
# interpretation is:
# if p(>|z|) in (0,0.001]
# (0,0.001] then ***
# (0.001,0.1] then **
# (0.1,1] then *
# 0.05 is alpha value.
# Since all values are in (0,0.001], we see '***' next to every Pr(|>z|)value.

# Dispersion parameter indicates whether the distribution is wide or narrow.
# For the binomial family(as specified in input), this value is 1.

# Next we have the values, 'Null deviance' and "Residual deviance". There is a
# significant difference between the values(143.47) which is an indication that
# the model is a good fit.

# The "Akaike information criterion (AIC)" is another measure of model fit.The
# lower its value compared to other models, the better the fit. Although
```

```
# individually, it is not that significant.

# Finally, fisher scoring iterations (5) is just a measure of how long it took
# to fit the model.
```

Problem 4 (19 points)

Problem 4.a. (9 points)

- Add a third predictor to the final model from **problem 3**, perform a likelihood ratio test to compare both models and report the p-value for the test.
- Is there any support for the additional term?
- Plot a ROC curve for both models and report the AUC, explain the results as you would communicate it to a colleague with a medical background with a very little statistical knowledge.
- Print a summary and explain the results as you would communicate it to a colleague with a medical background with a very little statistical knowledge.

```
# Add a 3rd predictor (urea) to previous model.
diabetes_model2<-glm(diabetes~age+glucose+urea,
                     data = dx2,family = 'binomial')
# Model 1:
# Odds ratio for age and confidence interval
or.age <- exp(coef(diabetes_model1)[2])
ci.age <- exp(confint(diabetes_model1))[2, ]
```

```
Waiting for profiling to be done...
```

```
# Round Odds ratio for age and confidence interval to 2 decimal places
round(c(or.age, ci.age), 2)
```

```
  age  2.5 % 97.5 %
 1.05   1.03   1.07
```

```
# Odds ratio for glucose and confidence interval
or.glucose <- exp(coef(diabetes_model1)[3])
ci.glucose <- exp(confint(diabetes_model1))[3, ]
```

```
Waiting for profiling to be done...
```

```r
# Round Odds ratio and confidence interval to 2 decimal places
round(c(or.glucose, ci.glucose), 2)
```

```
glucose   2.5 %  97.5 %
   1.02    1.01    1.02
```

```r
# Model 2:
# Odds ratio for age and confidence interval
or.age <- exp(coef(diabetes_model2)[2])
ci.age <- exp(confint(diabetes_model2))[2, ]
```

```
Waiting for profiling to be done...
```

```r
# Round Odds ratio and confidence interval to 2 decimal places
round(c(or.age, ci.age), 2)
```

```
   age  2.5 % 97.5 %
  1.05   1.03   1.07
```

```r
# Odds ratio for glucose and confidence interval
or.glucose <- exp(coef(diabetes_model2)[3])
ci.glucose <- exp(confint(diabetes_model2))[3, ]
```

```
Waiting for profiling to be done...
```

```r
# Round Odds ratio and confidence interval to 2 decimal places
round(c(or.glucose, ci.glucose), 2)
```

```
glucose   2.5 %  97.5 %
   1.02    1.01    1.02
```

```r
# Odds ratio for urea and confidence interval
or.urea <- exp(coef(diabetes_model2)[4])
```

```r
ci.urea <- exp(confint(diabetes_model2))[4, ]
```

```
Waiting for profiling to be done...
```

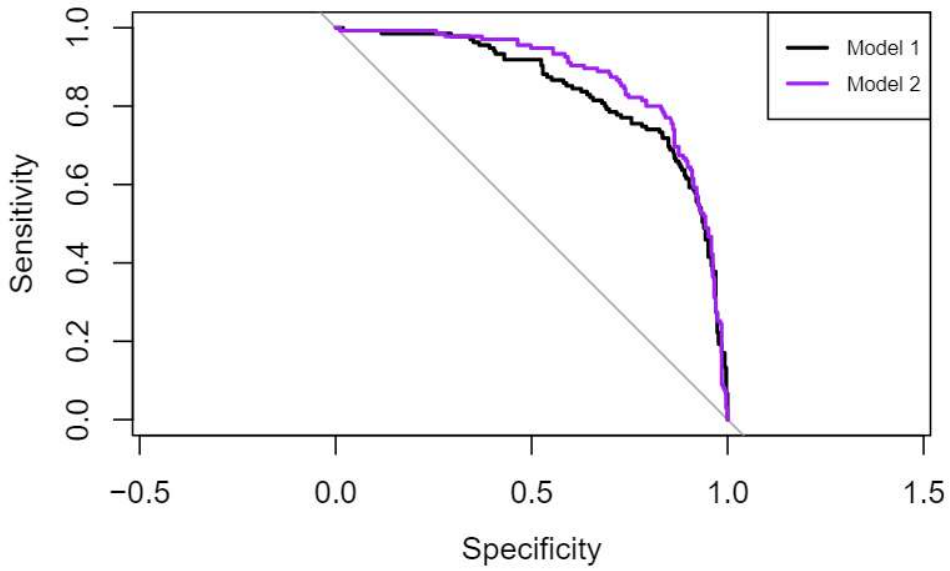
```r
# Round Odds ratio and confidence interval to 2 decimal places
round(c(or.urea, ci.urea), 2)
```

```
  urea  2.5 % 97.5 %
  1.01   1.01   1.02
```

```r
# Report the p value using pchisq()
p<-pchisq(diabetes_model1$deviance-diabetes_model2$deviance,
          df=1,lower.tail = FALSE)
cat('The p value is',p)
```

```
The p value is 3.822726e-06
```

```r
# Plot a roc curve to compare both models.
suppressMessages(invisible({
  roc(dx2$diabetes, diabetes_model1$fitted.values,
      plot = TRUE, xlim = c(0,1))
  # add = TRUE adds the roc curve to the previous plot
  roc(dx2$diabetes, diabetes_model2$fitted.values,
      plot = TRUE, xlim = c(0,1),
      add = TRUE, col = "purple")
  legend('topright',legend=c('Model 1','Model 2'),
       col=c('black','purple'),
       lty=c(1,1),lwd=c(2,2),cex = 0.7)
}))
```




```
# Print area under roc curve
cat('Area under roc curve',
    roc(dx2$diabetes, diabetes_model2$fitted.values,
      )$auc,'\n')
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

Area under roc curve 0.8745073
```

```
### ROC Curve explanation:
# The area under the roc curve is a good measure of the performance
# of a model. It is simply a plot between the False Positive Rate
# (Specificity) and True Positive Rate (Sensitivity). It is used
# to test performance of classification models in Machine Learning.
# It can also be used to test how significant the addition of a
# predictor variable is to the performance of model.
# As you can see below, there are 2 roc curves. The black one is
# that of model 1 and the purple one is that of model 2. Model 2
# is model 1 with an additional predictor variable (urea). Since
# auc did improve, urea is indeed a valid addition and there is
```

```r
# clearly support to it.

# Examine summary of the optimum model.
summary(diabetes_model2)
```

```
Call:
glm(formula = diabetes ~ age + glucose + urea, family = "binomial",
    data = dx2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0576  -0.6495  -0.3912   0.6083   2.9107

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.590163   0.707935  -9.309  < 2e-16 ***
age          0.047483   0.009922   4.786 1.70e-06 ***
glucose      0.017042   0.002472   6.894 5.42e-12 ***
urea         0.012717   0.002959   4.298 1.73e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 511.49  on 399  degrees of freedom
Residual deviance: 346.67  on 396  degrees of freedom
AIC: 354.67

Number of Fisher Scoring iterations: 5
```

```r
### Summary Explanation:
# The first line indicates the logistic regression model with the parameters
# that were inputted.

# The second line contains "deviance residuals" that are simply a 5-number
# summary of the residuals (just like in box-plot). We have minimum and
# maximum values that are roughly the same distance from origin if you think
# about it. We also see quantiles 1 followed by median and quantile 3 at last.

```

```
# Third line has the coefficients! The intercepts are values when the
# features are at 0. The estimates give expected change in response due
# to a unit change in the feature. Next we have standard errors that are
# simply errors involved in calculating estimates that allow to construct
# confidence intervals for that feature.

# Fourth line has the Z test values that describe how far the estimated value
# value is from null hypothesis value.

# Next we have the "p-values" (p(>|z|)). This is the probability of observing
# a value as extreme as z. We reject the null hypothesis if this value is
# very low. These probabilities follow Significance Codes below. The
# interpretation is:
# if p(>|z|) in (0,0.001]
# (0,0.001] then ***
# (0.001,0.1] then **
# (0.1,1] then *
# 0.05 is alpha value.
# Since all values are in (0,0.001], we see '***' next to every Pr(|>z|)value.

# Dispersion parameter indicates whether the distribution is wide or narrow.
# For the binomial family(as specified in input), this value is 1.

# Next we have the values, 'Null deviance' and "Residual deviance". There is a
# significant difference between the values(164.82) which is an indication that
# the model is a good fit.

# The "Akaike information criterion (AIC)" is another measure of model fit.The
# lower its value compared to other models, the better the fit. Although
# individually, it is not that significant.

# Finally, fisher scoring iterations (5) is just a measure of how long it took
# to fit the model.
```

Problem 4.b (10 points)

- Perform 10-folds cross validation for your chosen model based on the above answers.
- Report the mean cross-validated AUCs in 3 significant figures.

```r
# Initialize number of cross validation folds.
num.folds<-10
# Next we create 10 cross validation folds using the
# createFolds() function.
folds <- createFolds(dx2$diabetes, k = num.folds)
# Initialize predictor variable.
pred.cv <- NULL
# Train object to store observed training values.
regr.cv<-NULL
auc.cv <- numeric(num.folds)
for(f in 1:num.folds){
  # For each fold:
  # Store the test values
  test.idx <- folds[[f]]
  # Use setdiff() to exclude the k-fold rows from
  # original data and thus form train data
  train.idx <- setdiff(1:nrow(dx2), folds[[f]])
  # The regr.cv id the logistic regression object
  regr.cv[[f]] <- glm(diabetes~age+glucose+urea,
                      data = dx2,subset = train.idx,
                      family = 'binomial')
  # Now we use the predict() function to get prediction
  # values.
  pred.cv[[f]] <- data.frame(obs = dx2$diabetes[test.idx]
                             ,pred = predict(regr.cv[[f]],
                                             newdata = dx2,
                                             type = "response")[test.idx])
  # Store area under curve for the fold
  auc.cv[f] <- roc(obs ~ pred, data = pred.cv[[f]])$auc

}
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1
```

```
Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases

Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```r
# Mean cross-validated AUC is simply mean of auc.cv vector.
# Round resultan to 3 decimal places:
cat('The mean of aucs is',round(mean(auc.cv),3))
```

```
The mean of aucs is 0.867
```