Assignment 1

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

Pablo Ruiz Barnada

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 prvgecise 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()`.

```
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment1")
longegfr1 <- fread("longegfr1.csv")
longegfr2 <- fread("longegfr2.csv")

# Transform to data table
longegfr1.dt <- data.table(longegfr1, keep.rownames = TRUE)
longegfr2.dt <- data.table(longegfr2, keep.rownames = TRUE)

setnames(longegfr1.dt, "id", "ID")

# Merge data into single data table, ordering by subject identifier
# and follow-up time
dt <- merge(longegfr1.dt, longegfr2.dt, by = c("ID", "fu.years"),
            all = TRUE, sort = TRUE)

# Print first 10 values of new dataset
head(dt, 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.

```
# Creat empty arrays to store values
all.id <- c()
all.mean.egfr <- c()
all.futime <- c()
mean.futime <- c()

# Obtain the subset of observations for each patient, calculate mean eGFR
# and total follow-up time and store values
for (i in unique(dt[, ID])){
  subset.dt <- dt[dt[, ID == i]]
  mean.egfr <- mean(subset.dt[, egfr], na.rm = TRUE)
  fu.time <- tail(subset.dt[, fu.years], 1)

  all.id <- append(all.id, i)
  all.mean.egfr <- append(all.mean.egfr, mean.egfr)
  all.futime <- append(all.futime, fu.time)

  fu.time.patient <- c()
  for(j in 2:dim(subset.dt)[1]){
    time.diff <- subset.dt[j, fu.years] - subset.dt[(j-1), fu.years]
    fu.time.patient <- append(fu.time.patient, time.diff)
  }
  mean.fu.tim.pat <- mean(fu.time.patient, na.rm = TRUE)
  mean.futime <- append(mean.futime, mean.fu.tim.pat)
}

# Generte new data table with soterd values, update column names
dt.1b <- data.table(all.id, all.mean.egfr, all.futime, mean.futime)
colnames(dt.1b)[1] <- "ID"
colnames(dt.1b)[2] <- "Mean eGFR"
colnames(dt.1b)[3] <- "Total follow-up time (years)"
colnames(dt.1b)[4] <- "Mean follow-up time (years)"


```

Table 1: Number of patients in each eGFR range

eGFR range	Number of Patients
(0,15]	2
(15,30]	9
(30,60]	84
(60,90]	86
(90,175]	66

```r
# Print first 10 values
head(dt.1b, 10)
```

```
    ID Mean eGFR Total follow-up time (years) Mean follow-up time (years)
 1:  1  43.04333                       6.4586                    0.4613286
 2:  2  38.93294                       2.0698                    0.1293625
 3:  3  85.72000                       6.5161                    0.5012385
 4:  4  76.59308                       5.2786                    0.4398833
 5:  5  13.90892                       6.3929                    0.1360191
 6:  6  85.66435                       6.2313                    0.2832409
 7:  7  64.21758                       5.8453                    0.1826656
 8:  8  66.28333                       1.5606                    0.1950750
 9:  9  86.35750                       5.8700                    0.5336364
10: 10 107.00429                       5.1964                    0.8660667
```

```r
# Tabulation of number of patients
bins <- c(0,15,30,60,90,max(na.omit(dt[, "egfr"])))
tab.1b <- table(cut(as.numeric(unlist(na.omit(dt.1b[, "Mean eGFR"]))), bins))

# Create table
kable(tab.1b, format = "latex",
                caption = "Number of patients in each eGFR range",
                col.names = c('eGFR range', 'Number of Patients'),
                align = "c")
```

```r
cat("Number of patients with missing average eGFR:",
            sum(is.na(dt.1b$`Mean eGFR`)))
```

```
Number of patients with missing average eGFR: 3
```

Problem 1.c (6 points)

- For patients with average eGFR in the $(90, \texttt{max(eGFR)})$ 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.

```r
# Select patients with high eGFR
high.egfr.index <- which(dt.1b[, "Mean eGFR"] > 90)

# Create empty arrays to store values
all.id <- c()
all.sex <- c()
all.baseline.age <- c()
all.last.egfr <- c()
all.egfr.measure <- c()

# For each patient with high eGFR, collect required information
for (i in high.egfr.index){
  subset.dt <- dt[dt[, ID == i]]
  sex <- subset.dt[1, sex]
  baseline.age <- subset.dt[1, baseline.age]
  last.egfr <- tail(subset.dt[, fu.years], 1)
  egfr.measure <- dim(subset.dt)[1] - sum(is.na(subset.dt[, egfr]))

  all.id <- append(all.id, i)
  all.sex <- append(all.sex, sex)
  all.baseline.age <- append(all.baseline.age, baseline.age)
  all.last.egfr <- append(all.last.egfr, last.egfr)
  all.egfr.measure <- append(all.egfr.measure, egfr.measure)
}

# Generate data table qith required information, update column names
dt.1c <- data.table(all.id, all.sex, all.baseline.age,
                    all.last.egfr, all.egfr.measure,
                    dt.1b[high.egfr.index, "Mean eGFR"])
colnames(dt.1c)[1] <- "ID"
colnames(dt.1c)[2] <- "Sex"
colnames(dt.1c)[3] <- "Baseline Age"
colnames(dt.1c)[4] <- "Follow-up Years"
colnames(dt.1c)[5] <- "No. Measurements"
colnames(dt.1c)[6] <- "Mean eGFR"
```

```r

# Print summary of dataset
summary(dt.1c)
```

```
       ID              Sex          Baseline Age   Follow-up Years
 Min.   : 10.00   Min.   :0.0000   Min.   :22.10   Min.   :0.000
 1st Qu.: 86.25   1st Qu.:0.0000   1st Qu.:47.20   1st Qu.:1.671
 Median :144.00   Median :0.0000   Median :55.20   Median :4.857
 Mean   :141.88   Mean   :0.3333   Mean   :55.27   Mean   :4.064
 3rd Qu.:197.50   3rd Qu.:1.0000   3rd Qu.:63.80   3rd Qu.:5.937
 Max.   :250.00   Max.   :1.0000   Max.   :90.90   Max.   :6.590
 No. Measurements   Mean eGFR
 Min.   : 1.00    Min.   : 90.04
 1st Qu.: 5.00    1st Qu.: 99.13
 Median : 8.00    Median :109.81
 Mean   :11.91    Mean   :112.13
 3rd Qu.:13.75    3rd Qu.:123.20
 Max.   :57.00    Max.   :147.69
```

```r
cat("Proportion of women over total dataset:", length(which(dt.1c$Sex == 0))/dim(dt.1c)[1]
```

```
Proportion of women over total dataset: 0.6666667
```

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
# Set the layout to show the graphs properly
layout(matrix(1:4, nrow = 2, ncol = 2, byrow = TRUE), heights = lcm(0.5))
```

```r
par(mfrow=c(2,2), pin = c(2,1), mar = c(4,4,2,2))

# Specify the patients to show
patients <- c(3, 37, 162, 223)

# Obtain maximum fu years and eGFR to set the proper axis limits in the graph
max_fu.years <- max(dt[ID %in% patients, "fu.years"])
max_egfr <- max(dt[ID %in% patients, "egfr"], na.rm = TRUE)

# For each patient, plot required data
for (i in patients){
  subset.dt <- dt[dt[, ID == i]]

  plot(unlist(subset.dt[, "fu.years"]),
       unlist(subset.dt[, "egfr"]),
       xlim = c(0, as.numeric(ceiling(max_fu.years))),
       # ylim = c(0, as.numeric(ceiling(max_egfr)) + 10),
       ylim = c(0, 200),
       main = paste("Patient", i),
       sub = "Evolution of eGFR over time",
       xlab = "Time (Years)",
       ylab = "eGFR",
       pch = 20
  )

  legend(x = "topright",
         legend = c("Measured eGFR", "Regression", "Reg. w/out extreme val."),
         col = c("black", "blue", "red"),
         pch = c(20, NA, NA), lty= c(NA, 1,1),
         cex = 0.4)
  regr <- lm(unlist(subset.dt[, "egfr"]) ~ unlist(subset.dt[, "fu.years"]))
  abline(regr, col = "blue")

  max.pat.egfr <- which(subset.dt[, "egfr"] == max(subset.dt[, "egfr"],
                                                   na.rm = TRUE))
  min.pat.egfr <- which(subset.dt[, "egfr"] == min(subset.dt[, "egfr"],
                                                   na.rm = TRUE))

  regr2 <- lm(unlist(subset.dt[-c(min.pat.egfr,max.pat.egfr), "egfr"]) ~
                unlist(subset.dt[-c(min.pat.egfr,max.pat.egfr), "fu.years"]))
  abline(regr2, col = "red")
```

Table 2: 95% CI in Regression for Patient 3

Coefficient	2.5 %	97.5 %
Intercept	50.624	98.217
Beta1	-3.151	12.256

Table 3: 95% CI in Regression for Patient 37

Coefficient	2.5 %	97.5 %
Intercept	26.912	49.553
Beta1	-3.596	2.379

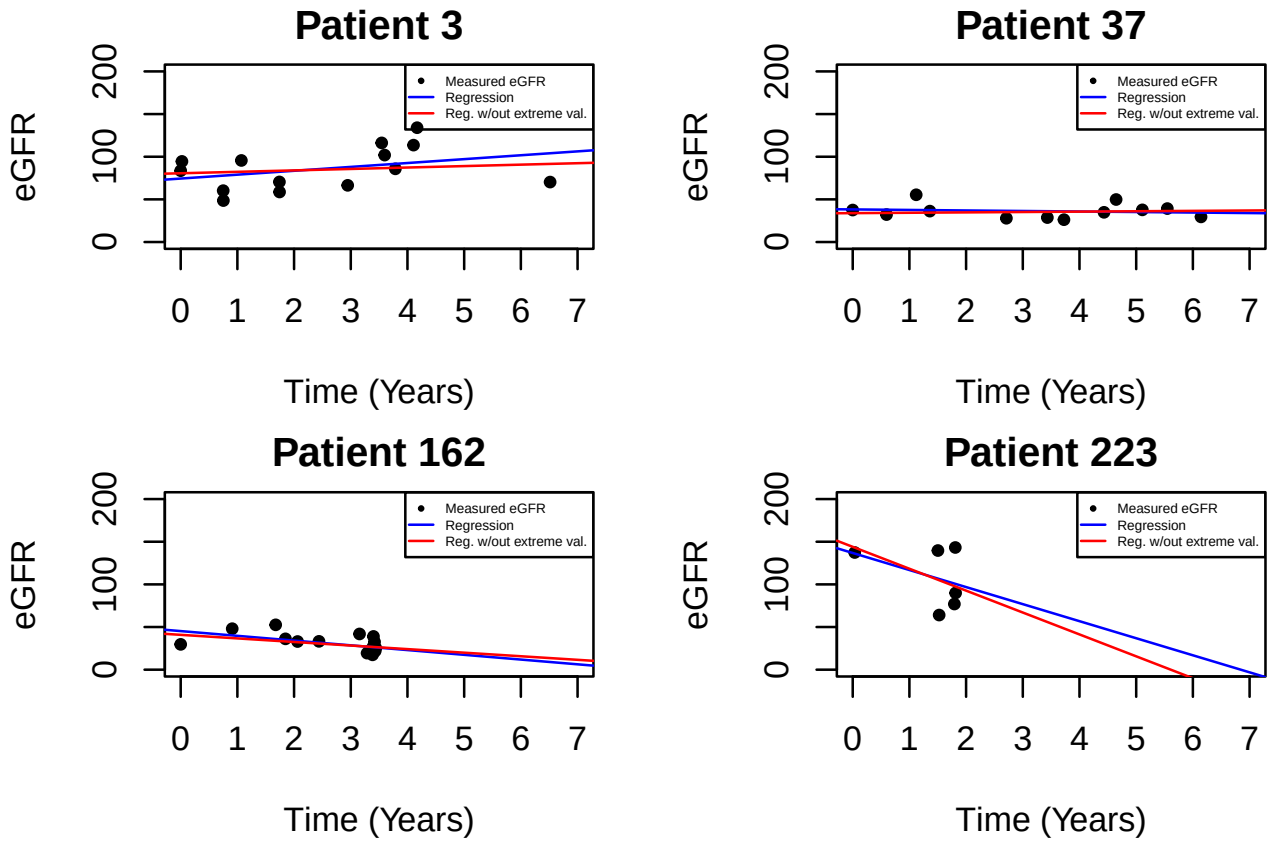
```
    ci.table <- as.data.table(confint(regr))
    ci.table <- cbind(c("Intercept", "Beta1"), ci.table)
    rownames(ci.table) <- c("Intercept", "\beta_1")
    tab <- kable(ci.table,
                 digits = 3,
                 format = "latex",
                 caption = paste("95\\% CI in Regression for Patient ", i),
                 col.names = c('Coefficient','2.5 %', '97.5 %'),
                 align = "c")

     # kable_styling(tab, bootstrap_options = c("bordered", "hover"))
     print(tab)
     cat("\n")
}
```

Table 4: 95% CI in Regression for Patient 162

Coefficient	2.5 %	97.5 %
Intercept	34.109	56.382
Beta1	-9.258	-1.872

Table 5: 95% CI in Regression for Patient 223

Coefficient	2.5 %	97.5 %
Intercept	34.718	238.864
Beta1	-85.938	45.966



The plots above show the measurements taken to patients 3, 37, 162 and 223. The black dots represent the eGFR measures over time, the blue line is a linear regression performed with each patient's data, and the red line is another regression where we omitted the largest and smallest eGFR measurements for each patient. In general, we can see a decreasing trend for all patients: after they start being checked regularly by a doctor (participating in the study), their eGFR is reduced. This happens for all patients except number 3, who is not able to reduce its eGFR and, in fact, slowly increases it.

In general, we see that when the extremes values are removed, the regression line is flatter. This makes sense as removing these datapoints makes the remaining observations lie in a thinner interval. However, patient 223, who has only 6 observations of eGFR, by removing

the two most extreme ones, obtains a regression line that is steeper. This is directly affected by the low number of data points used in the regression.

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.

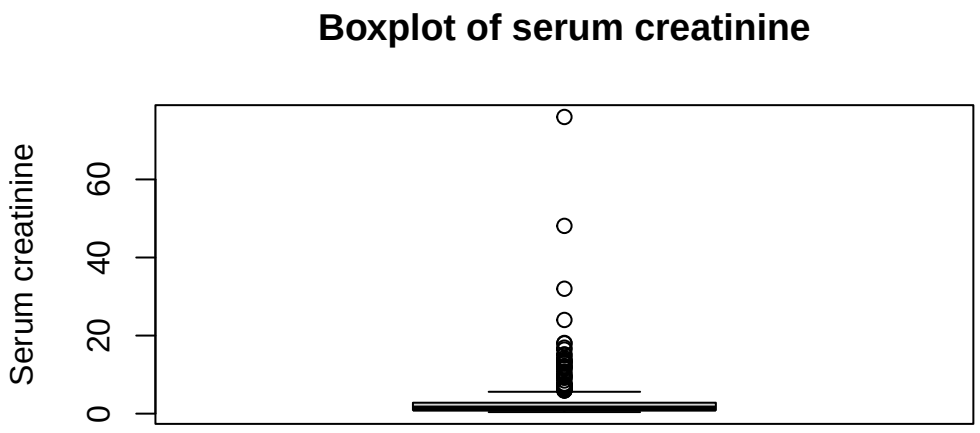
```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment1")
scr <- fread("scr.csv")
scr <- data.table(scr, keep.rownames = TRUE)
head(scr)
```

```
   age scr    sex ethnic
1:  48 1.2 Female  Other
2:   7 0.8   Male  Black
3:  62 1.8 Female   <NA>
4:  48 3.8 Female  Other
5:  51 1.4   Male  Other
6:  60 1.1   Male  Other
```

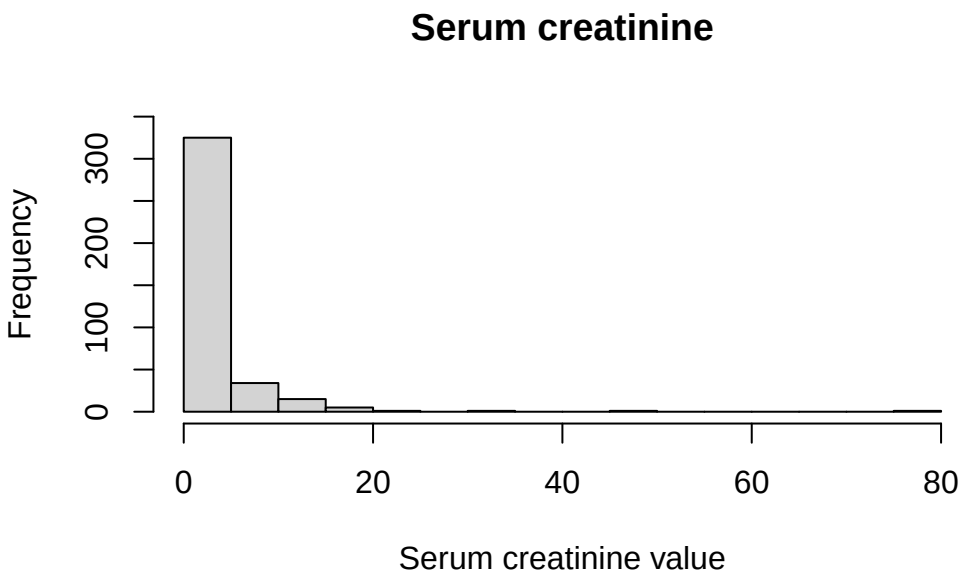
```r
summary(scr)
```

```
      age              scr              sex                ethnic
 Min.   : 2.00   Min.   : 0.400   Length:401         Length:401
 1st Qu.:42.00   1st Qu.: 0.900   Class :character   Class :character
 Median :55.00   Median : 1.300   Mode  :character   Mode  :character
 Mean   :51.48   Mean   : 3.072
 3rd Qu.:65.00   3rd Qu.: 2.800
 Max.   :83.00   Max.   :76.000
 NA's   :2       NA's   :18
```

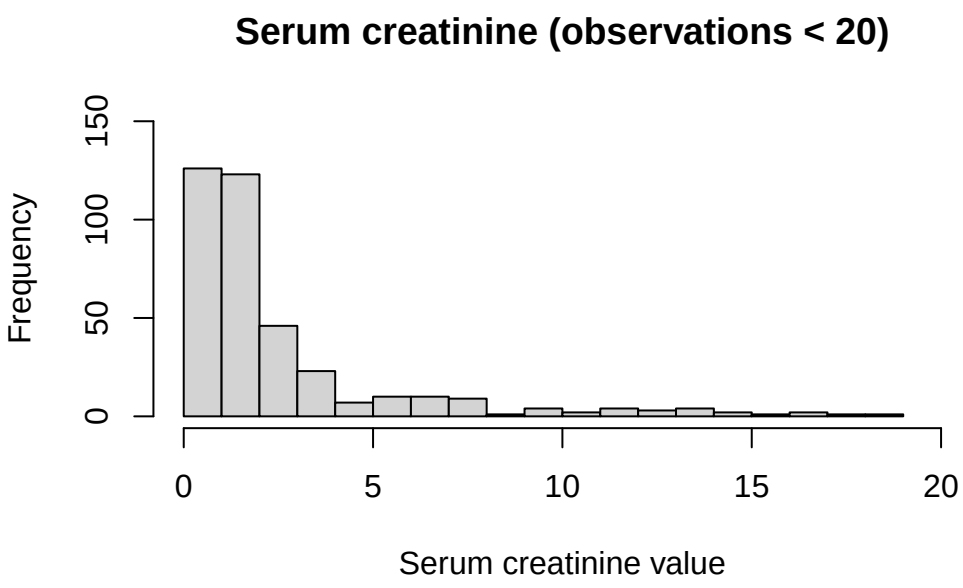
```r
# Examine distribution of serum creatinine
boxplot(scr$scr, main = "Boxplot of serum creatinine", ylab = "Serum creatinine")
```



```r
hist(scr$scr, breaks = 20, main = "Serum creatinine",
     xlab = "Serum creatinine value",
     # xlim = c(0,max(scr$scr, na.rm = TRUE)),
     xlim = c(0,80),

     ylim = c(0, 350))
```




```r
hist(scr[scr < 20, scr], breaks = 20,
     main = "Serum creatinine (observations < 20)",
     xlab = "Serum creatinine value",
     xlim = c(0,20),
     ylim = c(0, 150))
```



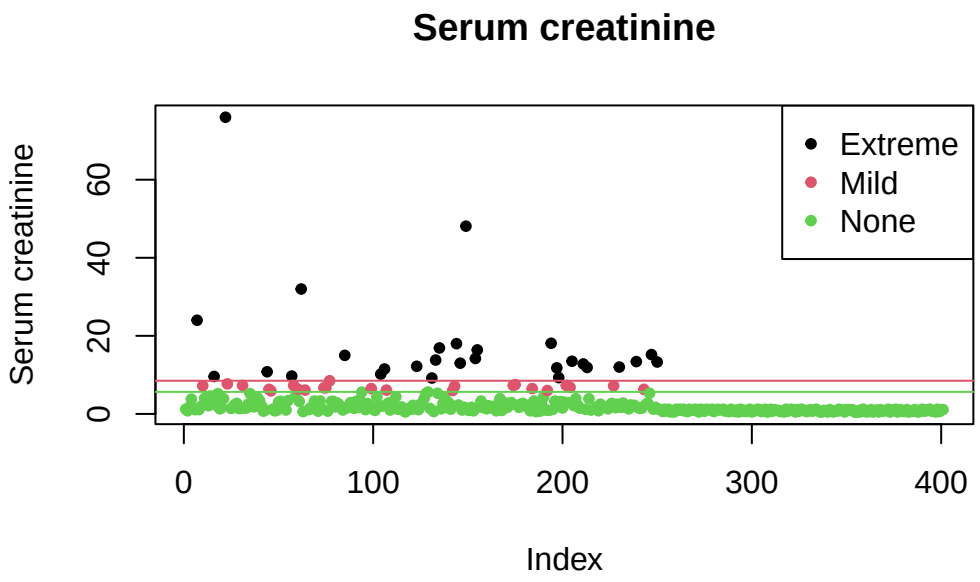

```r
# Calculate inter-quartile range and display it
iqr_lims <- quantile(scr$scr, c(0.25, 0.75), na.rm = TRUE) # interquartile range
iqr <- iqr_lims[2]-iqr_lims[1]

cat("The interquartile range is", iqr, "\nThe 25th percentile is", iqr_lims[1],
            "and the 75th percentile is", iqr_lims[2])
```

```
The interquartile range is 1.9
The 25th percentile is 0.9 and the 75th percentile is 2.8
```

From the above plots we can observe the distribution of serum creatinine levels in our dataset. We see that the interquartile range is small, only 1.9, with the 25th percentile being 0.9 and the 75th percentile being 2.8. Even though most observations lie within this short range, it can be observed in the boxplot that there are a few outliers that happen to be really far away from the confidence interval. By examining the first boxplot, we can confirm the previous observation. Most values lie in the range (0, 5], a few of them from (5, 10], and barely any have a value larger than 15. Moreover, by checking the second histogram (shows the distribution of values that are smaller than 20, only four observations are dropped in comparison to the previous histogram), we can confirm that most values lie in the range (0, 2], but no more than 50 observations in the dataset are larger than 4.

Now, we will analyse the possible outliers.

```r
# Possible mild outliers
lower.limit <- iqr_lims[1] - 1.5*iqr ### less than 0
upper.limit <- iqr_lims[2] + 1.5*iqr ### less than 0

# Possible extreme outliers
lower.limit.extreme <- iqr_lims[1] - 3*iqr
upper.limit.extreme <- iqr_lims[2] + 3*iqr

# Create new data table, set whether observations are mild outliers, extreme
# outliers or none.
scr <- cbind(scr, rep("None", nrow(scr)))
colnames(scr)[5] <- "outlier"

scr$outlier[scr$scr < lower.limit] <- "Mild"
scr$outlier[scr$scr > upper.limit] <- "Mild"

scr$outlier[scr$scr < lower.limit.extreme] <- "Extreme"
scr$outlier[scr$scr > upper.limit.extreme] <- "Extreme"

# Plot observations and possible outliers
plot(scr$scr, col = factor(scr$outlier),
     main = "Serum creatinine", pch = 20, ylab = "Serum creatinine")
legend(x = "topright", pch = 20,
       legend = levels(factor(scr$outlier)),
       col = c("black", "#DF536B", "#61D04F"))
# abline(a = lower.limit.extreme, b= 0, col = "red")
abline(a = upper.limit.extreme, b= 0,col = 2)
# abline(a = lower.limit, b= 0,col = "green")
abline(a = upper.limit, b= 0,col = 3)
```




```r
cat("The minimum serum creatine contained in the dataset is:",
    min(scr[, scr], na.rm = T),
    ".\nAn observation is considered a mild outlier
    if the observed value is larger than", upper.limit,
    ".\nAn observation is considered an extreme oulier
    if the observation is larger than", upper.limit.extreme, ".")
```

```
The minimum serum creatine contained in the dataset is: 0.4 .
An observation is considered a mild outlier
    if the observed value is larger than 5.65 .
An observation is considered an extreme oulier
    if the observation is larger than 8.5 .
```

In the above scatter plot we can observe all the serum creatinine points plotted, with red colour representing red outliers, and black points representing extreme outliers. We can see that there are a few extreme outliers, more than 25. With this information, we move on to see if these might be collected in the wrong units.

```r
# Calculate what the smallest datapoint in the dataset would be in mimcromol/dL
minval.molL <- min(scr[, scr], na.rm = T) * 88.42
scr <- as.data.frame(scr)
```

```r

# Change extreme values considered measured in wrong units
index.to.change <- which(scr$scr > minval.molL)
scr$scr[index.to.change] <- scr$scr[index.to.change]/88.42

scr <- as.data.table(scr)


#Update outlier values that might have changed
scr$outlier[scr$scr > lower.limit & scr$scr < upper.limit] <- "None"
```

The serum creatinine (SCR) levels in blood are indicative of kidney function. SCR is a waste product in the blood, and high levels are typically associated with malfunctioning of the kidneys and possible renal disease (eMedicineHealth). Normal ranges for SCR depend on age, sex, size and muscle mass, but the following can be taken as normal values for a healthy person, according to different sources (Mayo Clinic, Mount Sinai):

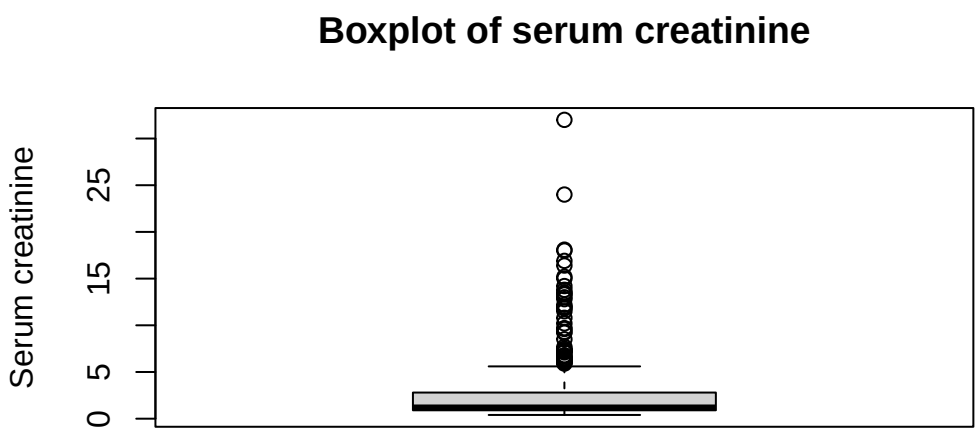
- Men: Approximately 0.7 - 1.3 mg/dL (~ 65.4 -119.3 µmol/L),
- Women: Approximately 0.6 - 1.1 mg/dL (~ 52.2 to 91.9 µmol/L).

SRC values above 1.3 mg/dL may be considered high, although for people with a single kidney it is normal to reach around 1.9 mg/dL in blood. Levels of SRC are associated with a risk for health when they surpass 5 mg/dL, for adults. In the case of infants, typical values are around 0.2 mg/dL, with risk to health being 2 mg/dL or more (eMedicineHealth). On the contrary, low levels of SCR are not directly related with kidney malfunctioning, but are associated with low muscle mass, low protein intake, aging, pregnancy, or even liver disease in extreme cases (NorthShore).

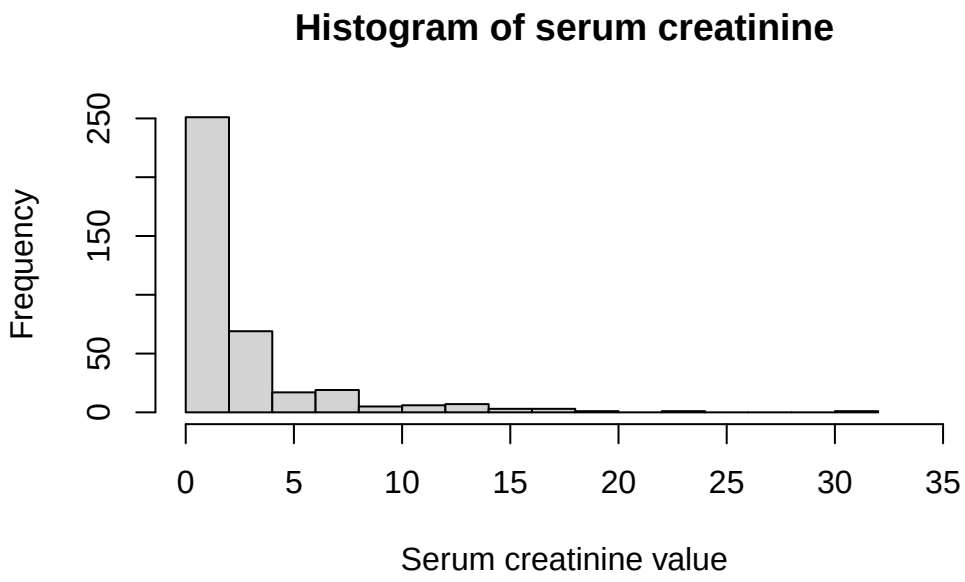
In our dataset, the minimum SCR observed is 0.4 mg/dL, which is a really low value. When this is transformed to mol/L units, the value obtained is 35.368 mol/L. By exploring our dataset, we observe that there are 2 observations that whose SRC is higher than 35.368 - these are assumed to be collected in mol/L, and are thus changed to mg/dL by dividing over 88.42. Aside from these, the following highest observations are 32 (corresponding to a 67 years old woman) and 24 (corresponding to a 68 years old man). These are extremely high values if measured in mg/dL, but extremely low if done in mol/L (0.36 and 0.27 mol/L respectively). Thus, it has been decided that these values are likely to have been collected with the correct units for patients with severe renal problems, and therefore no transformation is needed.

After the transformation, the new distributions can be observed below. As we only changed the two most extreme values, the overall distribution has not changed much. However, now all the observations are contained within a way smaller range.

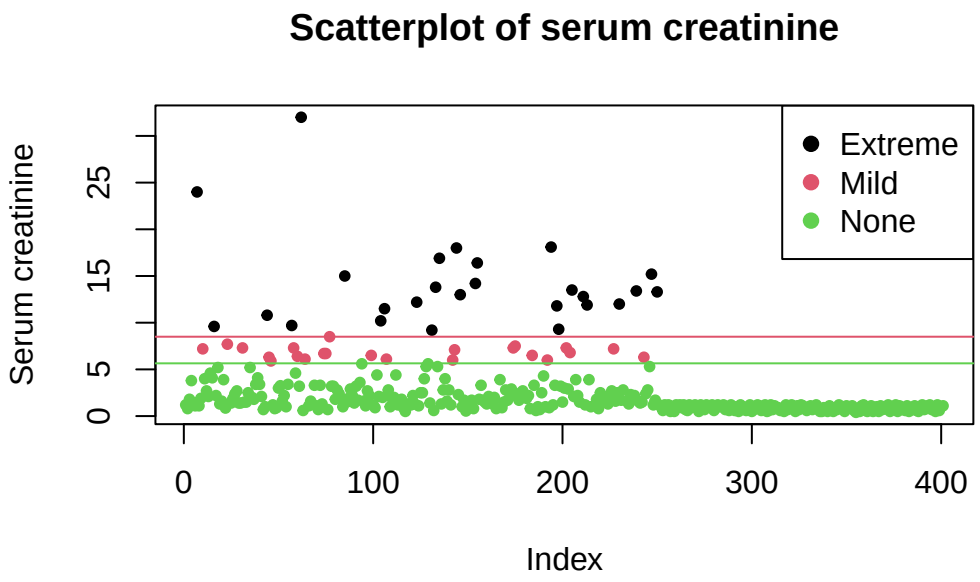
```r
# Examine new distribution
boxplot(scr$scr, main = "Boxplot of serum creatinine", ylab = "Serum creatinine")
```




```r
hist(scr$scr, breaks = 20, main = "Histogram of serum creatinine",
     xlab = "Serum creatinine value", xlim = c(0, 35))
```




```r
plot(scr$scr, col = factor(scr$outlier),
     main = "Scatterplot of serum creatinine",
     pch = 20,
     ylab = "Serum creatinine")
legend(x = "topright", pch = 19, legend = levels(factor(scr$outlier)),
       col = c("black", "#DF536B", "#61D04F"))
# abline(a = lower.limit.extreme, b= 0, col = "red")
abline(a = upper.limit.extreme, b= 0,col = 2)
# abline(a = lower.limit, b= 0,col = "green")
abline(a = upper.limit, b= 0,col = 3)
```



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()`.

```r
# Remove observations thta have missibng values, as the new measures require age,
# ethnicity, age and scr.
clean.scr <- na.omit(scr)

# Estimate eGFR with two different formulas
mdrd4 <- 175 * clean.scr$scr**(-1.154) *
  clean.scr$age**(-0.203) *
  if_else(clean.scr$sex == "Female", 0.742, 1) *
  if_else(clean.scr$ethnic == "Black", 1.212, 1)

ckd.epi <- 141 * if_else(clean.scr$sex=="Male",
                         pmin(clean.scr$scr/0.9, 1),
```

```
                          pmin(clean.scr$scr/0.7, 1)) **
  if_else(clean.scr$sex =="Male", -0.411, -0.329) *
  if_else(clean.scr$sex=="Male",
          pmax(clean.scr$scr/0.9, 1),
          pmax(clean.scr$scr/0.7, 1)) ** (-1.209) *
  0.993 ** (clean.scr$age) * if_else(clean.scr$sex == "Female", 1.018, 1) *
  if_else(clean.scr$ethnic == "Black", 1.159, 1)

# Create new data table, change colukn names
scr.2b <- cbind(clean.scr, mdrd4, ckd.epi)

colnames(scr.2b)[6] <- "MDRD4"
colnames(scr.2b)[7] <- "CKD-EPI"
```

First, we remove observations that have missing values. The dataset is large enough to remove these without problems, and for the eGFR calculation we need to have complete data for age, scr, sex and ethnicity.

```
cat("Mean of eGFR calculated via MDRD4:",
    round(mean(scr.2b$MDRD4, na.rm = TRUE),2))
```

```
Mean of eGFR calculated via MDRD4: 59.91
```

```
cat("\nStandard deviation of eGFR calculated via MDRD4:",
            round(sd(scr.2b$MDRD4, na.rm = TRUE), 2))
```

```
Standard deviation of eGFR calculated via MDRD4: 47.7
```

```
cat("\n\nMean of eGFR calculated via CKD-EPI:",
    round(mean(scr.2b$`CKD-EPI`, na.rm = TRUE),2))
```

```
Mean of eGFR calculated via CKD-EPI: 58.98
```

```
cat("\nStandard deviation of eGFR calculated via CKD-EPI:",
            round(sd(scr.2b$`CKD-EPI`, na.rm = TRUE), 2))
```

```
Standard deviation of eGFR calculated via CKD-EPI: 41.99
```

```
cat("\n\nPearson correlation coefficient between measures:",
            round(cor(scr.2b$MDRD4, scr.2b$`CKD-EPI`, method = "pearson"),2))
```

```
Pearson correlation coefficient between measures: 0.97
```

We can see that the correlation of both measures is really close to one. This makes sense as both MDRD4 and CKD-EPI are estimating the same quantity, eGFR.

```
cat("\n\nMean of eGFR calculated via MDRD4 in range [0, 60): ",
    round(mean(scr.2b[MDRD4 < 60, MDRD4],
                na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via MDRD4 in range [0, 60):  25.9
```

```
cat("\nStandard deviation of eGFR calculated via MDRD4 in range [0, 60):",
            round(sd(scr.2b[MDRD4 < 60, MDRD4],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via MDRD4 in range [0, 60): 17.3
```

```
cat("\nMean of eGFR calculated via CKD-EPI in range [0, 60): ",
    round(mean(scr.2b[MDRD4 < 60, `CKD-EPI`],
                na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via CKD-EPI in range [0, 60):  26.81
```

```r
cat("\nStandard deviation of eGFR calculated via CKD-EPI in range [0, 60):",
            round(sd(scr.2b[MDRD4 < 60, `CKD-EPI`],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via CKD-EPI in range [0, 60): 18.76
```

```r
cat("\nPearson correlation coefficient between measures in range [0, 60):",
            round(cor(scr.2b[MDRD4 < 60, MDRD4],
                      scr.2b[MDRD4 < 60, `CKD-EPI`],
                      method = "pearson"), 4))
```

```
Pearson correlation coefficient between measures in range [0, 60): 0.9966
```

```r
####################


cat("\n\nMean of eGFR calculated via MDRD4 in range [60, 90): ",
    round(mean(scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), MDRD4],
               na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via MDRD4 in range [60, 90):  73.41
```

```r
cat("\nStandard deviation of eGFR calculated via MDRD4 in range [60, 90):",
            round(sd(scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), MDRD4],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via MDRD4 in range [60, 90): 8.4
```

```r
cat("\nMean of eGFR calculated via CKD-EPI in range [60, 90): ",
    round(mean(scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), `CKD-EPI`],
               na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via CKD-EPI in range [60, 90):  80.18
```

```r
cat("\nStandard deviation of eGFR calculated via CKD-EPI in range [60, 90):",
            round(sd(scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), `CKD-EPI`],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via CKD-EPI in range [60, 90): 10.42
```

```r
cat("\nPearson correlation coefficient between measures in range [60, 90):",
            round(cor(scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), MDRD4],
                      scr.2b[(MDRD4 >= 60) & (MDRD4 < 90), `CKD-EPI`],
                      method = "pearson"), 2))
```

```
Pearson correlation coefficient between measures in range [60, 90): 0.93
```

```r
######################


cat("\n\nMean of eGFR calculated via MDRD4 in range (>= 90)): ",
    round(mean(scr.2b[(MDRD4 >= 90), MDRD4],
               na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via MDRD4 in range (>= 90)):  133.91
```

```r
cat("\nStandard deviation of eGFR calculated via MDRD4 in range (>= 90):",
            round(sd(scr.2b[(MDRD4 >= 90), MDRD4],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via MDRD4 in range (>= 90): 34.04
```

```r
cat("\nMean of eGFR calculated via CKD-EPI in range (>= 90): ",
    round(mean(scr.2b[(MDRD4 >= 90), `CKD-EPI`],
               na.rm = TRUE),
          2))
```

```
Mean of eGFR calculated via CKD-EPI in range (>= 90):  119.71
```

```r
cat("\nStandard deviation of eGFR calculated via CKD-EPI in range (>= 90):",
            round(sd(scr.2b[(MDRD4 >= 90), `CKD-EPI`],
                     na.rm = TRUE),
                  2))
```

```
Standard deviation of eGFR calculated via CKD-EPI in range (>= 90): 19.26
```

```r
cat("\nPearson correlation coefficient between measures in range [>= 90):",
            round(cor(scr.2b[(MDRD4 >= 90), MDRD4],
                      scr.2b[(MDRD4 >= 90), `CKD-EPI`],
                      method = "pearson"),2))
```

```
Pearson correlation coefficient between measures in range [>= 90): 0.84
```

When differentiating by range, we see that the estimations by MDRD4 and CKD-EPI are really close for smaller values, but they slightly break apart as they grow larger. In any case, the estimations are still similar, and their correlation coefficient is kept at 0.84.

In particular, it can be observed that MDRD4 tends to estimate smaller eGFR for values smaller than 90; however, for larger values, it tends to overestimate in comparison to CKD-EPI. The standard deviations are generally low for both methods in observations lower than 90, but they grow for larger than 90. This makes sense, as there are less overall estimations larger than 90.

```r
# Print first 15 values of new dataset
head(scr.2b, 15)
```

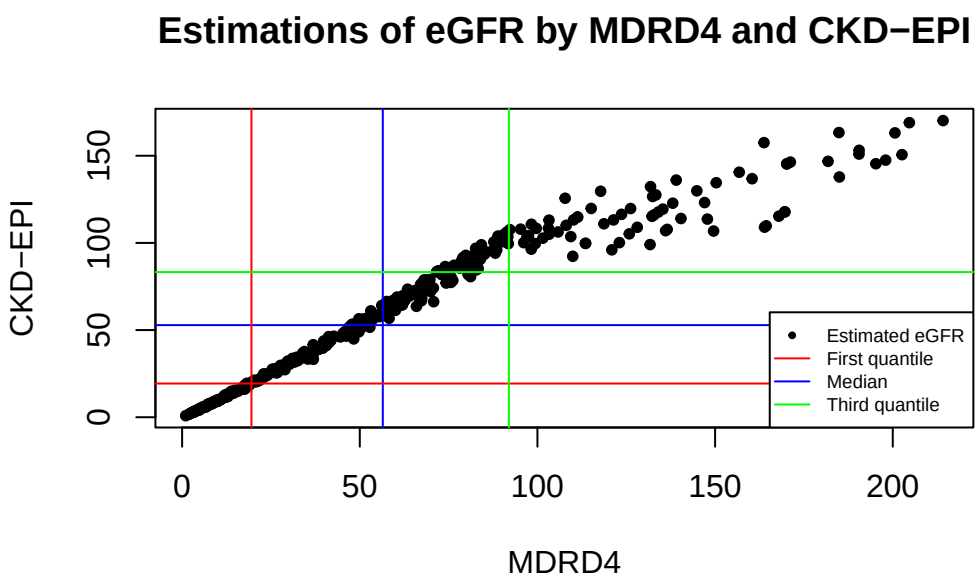
```
    age  scr    sex ethnic outlier      MDRD4    CKD-EPI
 1:  48  1.2 Female  Other    None  47.948478  53.397905
 2:   7  0.8   Male  Black    None 184.850199 163.294281
 3:  48  3.8 Female  Other    None  12.678846  13.252444
 4:  51  1.4   Male  Other    None  53.428078  57.761862
 5:  60  1.1   Male  Other    None  68.281993  72.578746
 6:  68 24.0   Male  Other Extreme   1.897907   1.651094
 7:  24  1.1   Male  Black    None  99.676006 108.322818
 8:  52  1.9 Female  Other    None  27.759499  29.787787
 9:  53  7.2   Male  Other    Mild   8.010292   7.864905
10:  50  4.0 Female  Other    None  11.851513  12.281808
11:  63  2.7   Male  Other    None  23.987121  23.998394
12:  68  2.1 Female  Other    None  23.420792  23.587189
13:  68  4.6   Male  Other    None  12.770744  12.166705
14:  68  4.1   Male  Black    None  17.676195  16.205967
15:  40  9.6   Male  Other Extreme   6.085256   6.085585
```

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
# Plot as requested
plot(scr.2b$MDRD4, scr.2b$`CKD-EPI`,
     main="Estimations of eGFR by MDRD4 and CKD-EPI",
     ylab = "CKD-EPI",
     xlab = "MDRD4",
     pch = 20)

abline(h = median(scr.2b$MDRD4), col = "blue")
abline(v = median(scr.2b$`CKD-EPI`),col = "blue")
```

```r

abline(h = quantile(scr.2b$MDRD4, probs = 0.25), col = "red")
abline(v = quantile(scr.2b$`CKD-EPI`, probs = 0.25), col = "red")

abline(h = quantile(scr.2b$MDRD4, probs = 0.75), col = "green")
abline(v = quantile(scr.2b$`CKD-EPI`, probs = 0.75), col = "green")

legend(x = "bottomright", pch = c(20, NA, NA, NA), lty = c(0,1,1,1),
       legend = c("Estimated eGFR", "First quantile", "Median", "Third quantile"),
       col = c("black", "red","blue", "green"), cex = 0.6, bg = "white")
```



The relationship is mostly linear, mainly for the lower estimated values. Moreover, the variance for the lower values is really small, while for larger estimations it increases. It is reasonable that the relation between both ways to estimate eGFR is linear as they are estimating the same quantity, and they should be consistent over the complete spectrum of possible values.

For lower values (estimated eGFR of less than 100), CKD-EPI tends to estimate slightly higher values than MDRD4. This tendency changes for eGFRs larger than 100, where MDRD4 usually makes higher estimations. This indeed makes sense according to the outputs from part 2.b.

It could also be argued that there is some kind of logarithmic relationship that would better fit larger values, but it would create more problems for the lower. This "relationship" for larger

values might just be due to the increased variance of having less observations, and under the influence of outliers that were not transformed before.

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()`.

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment1")
cohort <- fread("cohort.csv")
lab1 <- fread("lab1.csv")
linker <- fread("linker.csv")

# Merge data
cohort.dt <- merge(merge(cohort, linker, by = "id", all = TRUE), lab1,
                   by = "LABID", all = TRUE)
head(cohort.dt)
```

```
     LABID      id yob.x age bp diabetes albumin yob.y urea creatinine glucose
1:   LID_1 PID_285  1986  33 80        0   normo  1986 37.0    106.104     100
2:  LID_10 PID_153  1980  39 70        1   normo  1980 20.0     70.736     121
3: LID_100  PID_13  1951  68 70        1   micro  1951 72.0    185.682     208
4: LID_101 PID_110  1965  54 70        1    <NA>  1965 50.1    167.998     233
5: LID_102 PID_222  1953  66 70        1   micro  1953 30.0    150.314     248
6: LID_103 PID_103  2002  17 60        0   normo  2002 32.0    185.682      92
```

```r
# Sanity checks
length(intersect(cohort$id, cohort.dt$id))
```

```
[1] 400
```

```r
length(intersect(lab1$LABID, cohort.dt$LABID))
```

```
[1] 400
```

```r
length(intersect(linker$id, cohort.dt$id))
```

```
[1] 400
```

```r
length(intersect(linker$LABID, cohort.dt$LABID))
```

```
[1] 400
```

```r
n_distinct(cohort$id)
```

```
[1] 400
```

```r
n_distinct(cohort.dt$id)
```

```
[1] 400
```

```r
n_distinct(lab1$LABID)
```

```
[1] 400
```

```r
n_distinct(cohort.dt$LABID)
```

```
[1] 400
```

```r
all(cohort.dt$yob.x == cohort.dt$yob.y)
```

```
[1] TRUE
```

```r
#Drop unnecessary columns, update names
cohort.dt <- cohort.dt[, !c("LABID", "yob.y")]
colnames(cohort.dt)[2] <- "yob"

# Factor albumin
cohort.dt$albumin <- as.integer(factor(cohort.dt$albumin,
                                       levels = c("normo", "micro", "macro")))

head(cohort.dt, 10)
```

```
        id  yob age bp diabetes albumin urea creatinine glucose
 1: PID_285 1986  33 80        0       1 37.0    106.104     100
 2: PID_153 1980  39 70        1       1 20.0     70.736     121
 3:  PID_13 1951  68 70        1       2 72.0    185.682     208
 4: PID_110 1965  54 70        1      NA 50.1    167.998     233
```

```
 5: PID_222 1953  66 70        1       2 30.0    150.314     248
 6: PID_103 2002  17 60        0       1 32.0    185.682      92
 7: PID_200 1954  65 80        0       1 37.0    132.630      92
 8: PID_378 1955  64 70        0       1 27.0     61.894      97
 9: PID_267 1964  55 80        0       1 17.0    106.104     133
10: PID_271 1996  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
# Copy data, update age
cohort.dt.copy <- cohort.dt %>% copy()

cohort.dt.copy$age <- if_else(is.na(cohort.dt.copy$age),
                              as.numeric(format(Sys.Date(), "%Y")) -
                            cohort.dt.copy$yob,
                                      cohort.dt.copy$age )

# Sanity check
any(is.na(cohort.dt.copy$age))
```

```
[1] FALSE
```

```r
# Make function for mean imputation
impute.to.mean <- function(var){
  cohort.dt.copy[[var]][is.na(cohort.dt.copy[[var]])] <-
    mean(cohort.dt.copy[[var]], na.rm = TRUE)
  return(cohort.dt.copy)
}

# Run the dunction
continuous <- c("bp", "urea", "creatinine", "glucose")
for (i in continuous){
```

```
  cohort.dt.copy <- impute.to.mean(i)
}

# Sanity checks
any(is.na(cohort.dt.copy$bp))
```

```
[1] FALSE
```

```
any(is.na(cohort.dt.copy$urea))
```

```
[1] FALSE
```

```
any(is.na(cohort.dt.copy$creatinine))
```

```
[1] FALSE
```

```
any(is.na(cohort.dt.copy$glucose))
```

```
[1] FALSE
```

```
###############

# Make function for mode imputation
impute.to.mode <- function(var){
  uniq.val <- unique(cohort.dt.copy[[var]])
  mode <- uniq.val[which.max(tabulate(match(var, uniq.val)))]

  cohort.dt.copy[[var]][is.na(cohort.dt.copy[[var]])] <- mode

  return(cohort.dt.copy)
}


# Run function
categorical <- c("diabetes", "albumin")
for (i in categorical){
```

```
  cohort.dt.copy <- impute.to.mode(i)
}

# Sanity checks
any(is.na(cohort.dt.copy$diabetes))
```

```
[1] FALSE
```

```
any(is.na(cohort.dt.copy$albumin))
```

```
[1] FALSE
```

```
# Print first observations of dataset
head(cohort.dt.copy, 15)
```

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

We will analyse each variable one step at a time.

Numerical variables

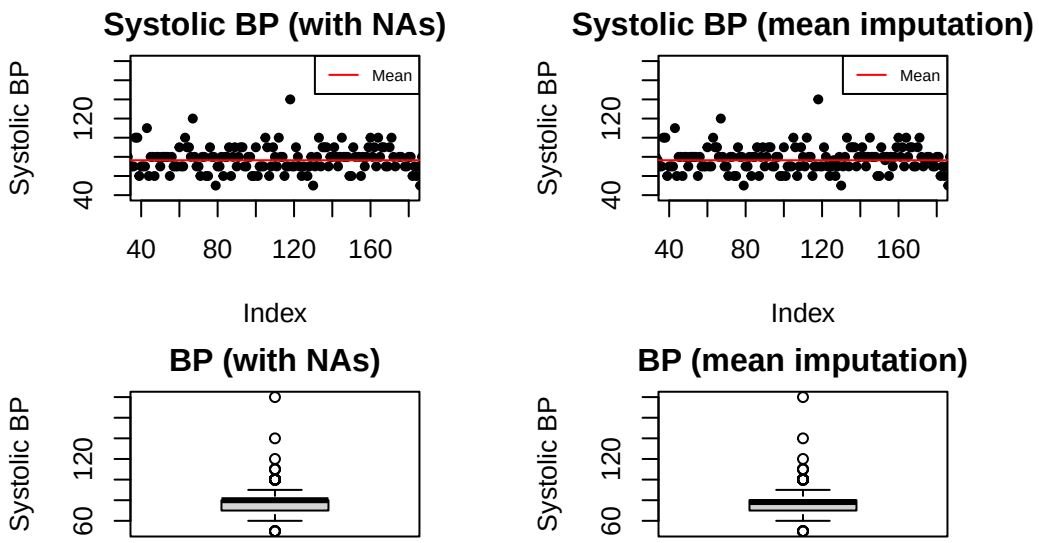
Firstly, we mention the case of age and date of birth. There are 9 rows in the dataset that are lacking the age of the patient. In all these, the year of birth is in a different format (decimal point) compared to other observations, in particular, the 9 observations show year of birth =

1967.517 . This number is precisely the mean year of birth of the rest of the observations in the dataset. Therefore, we assume there is a valid reason to input this value in what were probably missing observations. As a consequence, it is valid to calculate the age from these observations, and so we decide to proceed with mean imputation.

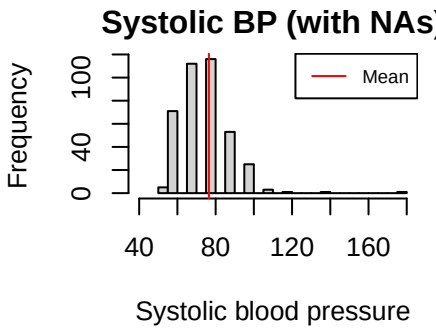
```
### BP
# Print summary information
cat("There are", sum(is.na(cohort.dt[["bp"]])), "NA values for blood pressure.")
```

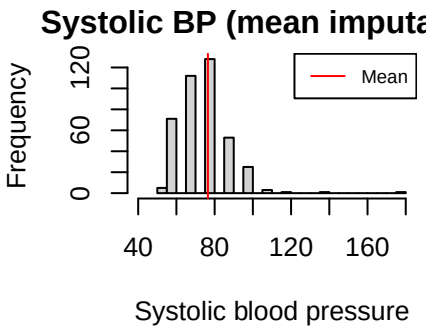
```
There are 12 NA values for blood pressure.
```

```
# Set layout
par(mfrow=c(2,2), mar = c(4,4,2,3.5))

# Plot data
plot(cohort.dt[["bp"]], ylab = "Systolic BP",
     main = "Systolic BP (with NAs)",pch = 20,
     xlim = c(40, max(cohort.dt[["bp"]], na.rm = TRUE)),
     ylim = c(40, max(cohort.dt[["bp"]], na.rm = TRUE)))

abline(h = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.6)

plot(cohort.dt.copy[["bp"]], ylab = "Systolic BP",
     main = "Systolic BP (mean imputation)",pch = 20,
    xlim = c(40, max(cohort.dt.copy[["bp"]], na.rm = TRUE)),
    ylim = c(40, max(cohort.dt.copy[["bp"]], na.rm = TRUE)))

abline(h = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.6)


boxplot(cohort.dt[["bp"]], main = "BP (with NAs)", ylab = "Systolic BP")
boxplot(cohort.dt.copy[["bp"]], main = "BP (mean imputation)",
        ylab = "Systolic BP")
```




```r
hist(cohort.dt[["bp"]], breaks = 20,
     main = "Systolic BP (with NAs)",
     xlim = c(40, max(cohort.dt[["bp"]], na.rm = TRUE)),
     xlab = "Systolic blood pressure")
abline(v = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)

hist(cohort.dt.copy[["bp"]], breaks = 20,
     main = "Systolic BP (mean imputation)",
     xlim = c(40, max(cohort.dt.copy[["bp"]], na.rm = TRUE)),
     xlab = "Systolic blood pressure")
abline(v = mean(cohort.dt[["bp"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)
```





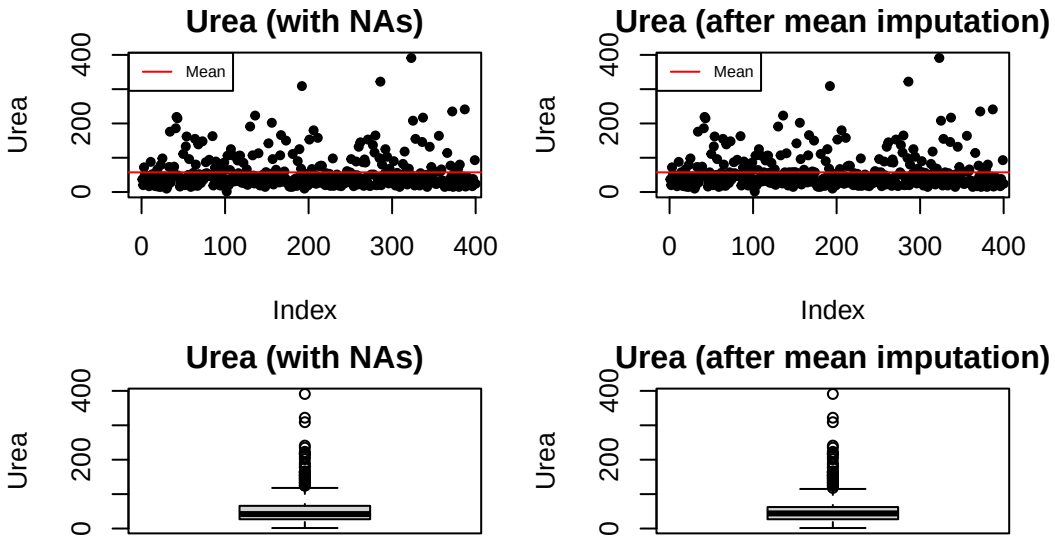
The first studied characteristic is systolic blood pressure (BP). As printed above, there are 12 missing values. From the above plots, it can be observed that the distribution of BP values and quartiles after mean imputation does not change, with a really slight reduction of the median. Thus, we decide to proceed with the mean input for blood pressure.

```
###Urea

cat("There are", sum(is.na(cohort.dt[["urea"]])), "NA values for urea.")
```

```
There are 19 NA values for urea.
```

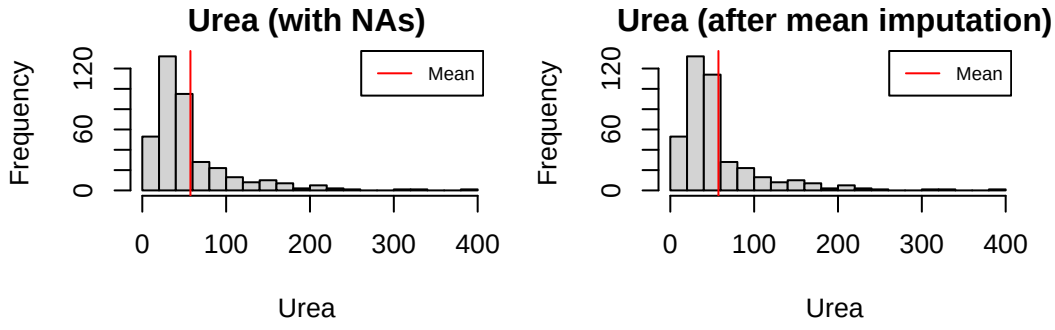
```
par(mfrow=c(2,2), mar = c(4,4,2,1.5))

plot(cohort.dt[["urea"]], ylab = "Urea", main = "Urea (with NAs)",
       xlim = c(0, max(cohort.dt[["urea"]], na.rm = TRUE)),pch = 20,
       ylim = c(0, max(cohort.dt[["urea"]], na.rm = TRUE)))
  abline(h = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
  # abline(v = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
legend(x = "topleft", lty = 1, col = "red", legend = "Mean", cex = 0.6)

plot(cohort.dt.copy[["urea"]], ylab = "Urea",
```

```
      main = "Urea (after mean imputation)",pch = 20,
      xlim = c(0, max(cohort.dt.copy[["urea"]], na.rm = TRUE)),
      ylim = c(0, max(cohort.dt.copy[["urea"]], na.rm = TRUE)))

abline(h = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
legend(x = "topleft", lty = 1, col = "red", legend = "Mean", cex = 0.6)

boxplot(cohort.dt[["urea"]], main = "Urea (with NAs)", ylab = "Urea")
boxplot(cohort.dt.copy[["urea"]],
        main = "Urea (after mean imputation)", ylab = "Urea")
```




```
hist(cohort.dt[["urea"]], breaks = 20, main = "Urea (with NAs)",
     xlim = c(min(cohort.dt[["urea"]], na.rm = TRUE),
              max(cohort.dt[["urea"]], na.rm = TRUE)),
     xlab = "Urea")
abline(v = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)

hist(cohort.dt.copy[["urea"]], breaks = 20,
     main = "Urea (after mean imputation)",
```

```
      xlim = c(min(cohort.dt.copy[["urea"]], na.rm = TRUE),
               max(cohort.dt.copy[["urea"]], na.rm = TRUE)),
      xlab = "Urea")
abline(v = mean(cohort.dt[["urea"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)
```



Now we focus on urea values, which have 19 missing observations. The distribution after doing mean imputation is kept similar, with most values contained in a small IQR. The imputed values correspond to the second largest bin in the histogram, which is close the largest one. Thus, the imputed values seem to be a good approximation of reality (even if not ideal), so we decide to proceed with mean imputation.

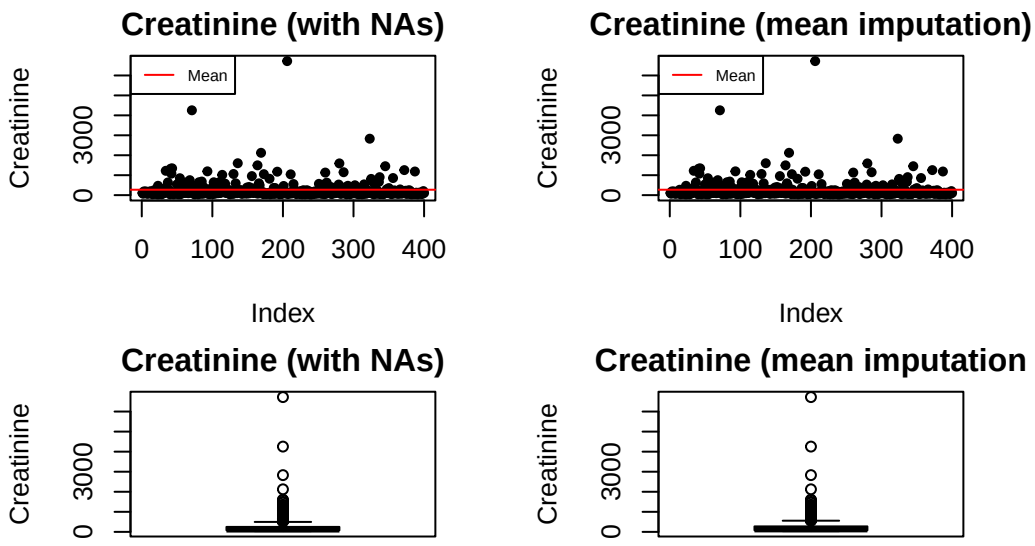
```
### Creatinine

cat("There are", sum(is.na(cohort.dt[["creatinine"]])),
    "NA values for creatinine.")
```

```
There are 17 NA values for creatinine.
```

```r
cat("Median:", median(cohort.dt[["creatinine"]], na.rm = T), "\nMean:",
    mean(cohort.dt[["creatinine"]], na.rm = T))
```

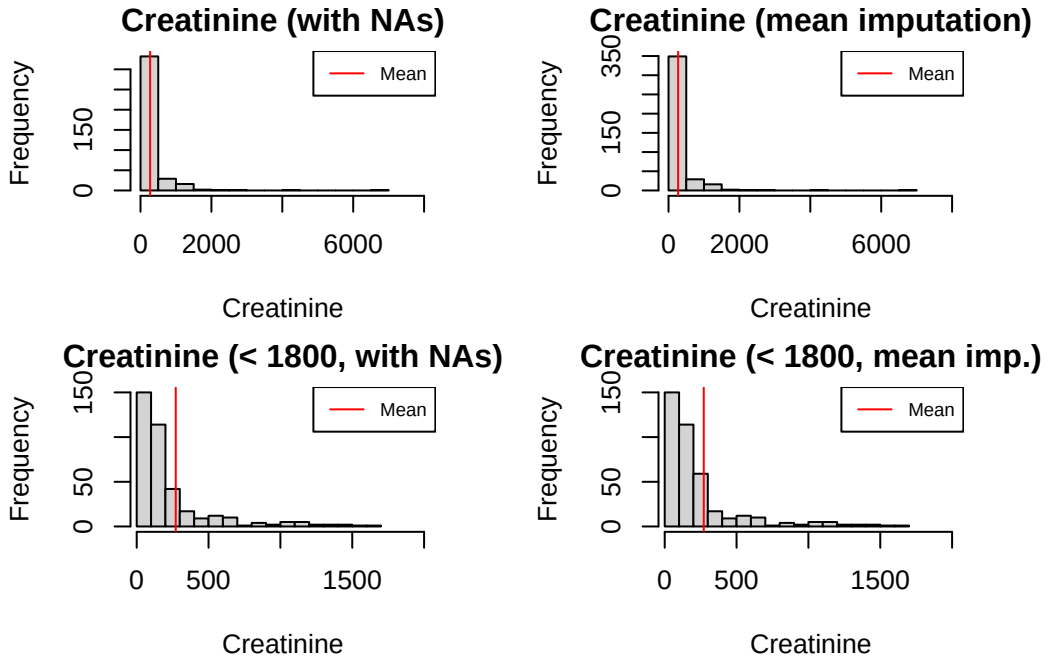
```
Median: 114.946
Mean: 271.6664
```

```r
par(mfrow=c(2,2), mar = c(4,4,2,3))

plot(cohort.dt[["creatinine"]], ylab = "Creatinine",
     main = "Creatinine (with NAs)",pch = 20,
     xlim = c(0, 400),
     ylim = c(0, max(cohort.dt[["creatinine"]], na.rm = TRUE)))
abline(h = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
legend(x = "topleft", lty = 1, col = "red", legend = "Mean", cex = 0.6)

plot(cohort.dt.copy[["creatinine"]], ylab = "Creatinine",
     main = "Creatinine (mean imputation)",pch = 20,
    xlim = c(0, 400),
    ylim = c(0, max(cohort.dt.copy[["creatinine"]], na.rm = TRUE)))
abline(h = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
legend(x = "topleft", lty = 1, col = "red", legend = "Mean", cex = 0.6)

boxplot(cohort.dt[["creatinine"]], main = "Creatinine (with NAs)",
        ylab = "Creatinine")
boxplot(cohort.dt.copy[["creatinine"]],
        main = "Creatinine (mean imputation", ylab = "Creatinine")
```




```r
hist(cohort.dt[["creatinine"]], breaks = 20,
     main = "Creatinine (with NAs)",
     xlim = c(min(cohort.dt[["creatinine"]], na.rm = TRUE),
              8000),
     xlab = "Creatinine")
abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)

hist(cohort.dt.copy[["creatinine"]], breaks = 20,
     main = "Creatinine (mean imputation)",
     xlim = c(min(cohort.dt.copy[["creatinine"]], na.rm = TRUE),
              8000),
     xlab = "Creatinine")
abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)


##################################

hist(cohort.dt[creatinine < 1800, creatinine], breaks = 20,
     main = "Creatinine (< 1800, with NAs)",
     xlim = c(min(cohort.dt[["creatinine"]], na.rm = TRUE),
```

```
                2000),
     xlab = "Creatinine")
abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)

hist(cohort.dt.copy[creatinine < 1800, creatinine], breaks = 20,
     main = "Creatinine (< 1800, mean imp.)",
     xlim = c(min(cohort.dt.copy[["creatinine"]], na.rm = TRUE),
                2000),
     xlab = "Creatinine")
abline(v = mean(cohort.dt[["creatinine"]], na.rm = TRUE),
       col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7)
```



Next, we study creatinine. There are 17 missing values for this characteristic. It can be observed that there are a few really big outliers, which directly affect the mean. This provokes that the mean is a lot larger than the median, and the inputted values are not that close to the median value. This can be particularly observed in the second pair of histograms, were we only plotted the creatinine values that are smaller than 1800. Therefore, we decide to not use mean imputation for creatinine values.

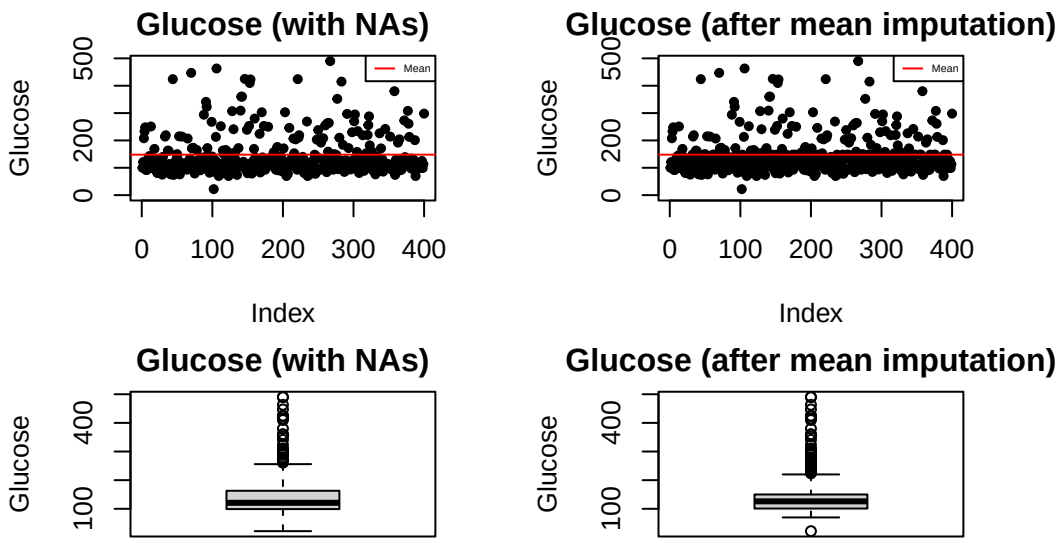
```r
### Glucose
cat("There are", sum(is.na(cohort.dt[["glucose"]])), "NA values for glucose")
```

```
There are 44 NA values for glucose
```

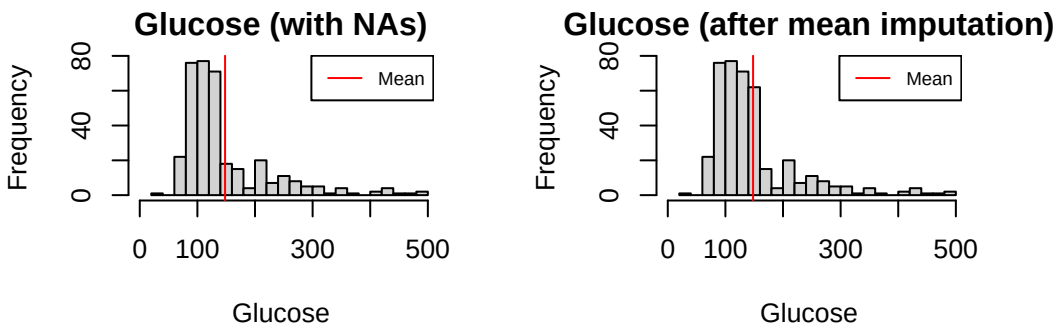
```r
cat("\nMedian:", median(cohort.dt[["glucose"]], na.rm = T), "\nMean:",
    mean(cohort.dt[["glucose"]], na.rm = T))
```

```
Median: 121
Mean: 148.0365
```

```r
par(mfrow=c(2,2), mar = c(4,4,2,3))

plot(cohort.dt[["glucose"]], ylab = "Glucose",
     main = "Glucose (with NAs)", pch = 20,
      xlim = c(0, 400),
      ylim = c(0, max(cohort.dt[["glucose"]], na.rm = TRUE)))
abline(h = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.4,
       bg = "white")

plot(cohort.dt.copy[["glucose"]], ylab = "Glucose",
     main = "Glucose (after mean imputation)", pch = 20,
     xlim = c(0, 400),
     ylim = c(0, max(cohort.dt.copy[["glucose"]], na.rm = TRUE)))
abline(h = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
# abline(v = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.4,
       bg = "white")
boxplot(cohort.dt[["glucose"]], main = "Glucose (with NAs)", ylab = "Glucose")
boxplot(cohort.dt.copy[["glucose"]],
        main = "Glucose (after mean imputation)", ylab = "Glucose")
```




```r
hist(cohort.dt[["glucose"]], breaks = 18, main = "Glucose (with NAs)",
    xlim = c(0,
             max(cohort.dt[["glucose"]], na.rm = TRUE)),
    xlab = "Glucose")
abline(v = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7,
       bg = "white")
hist(cohort.dt.copy[["glucose"]], breaks = 18,
     main = "Glucose (after mean imputation)",
     xlim = c(0,
              max(cohort.dt.copy[["glucose"]], na.rm = TRUE)),
     xlab = "Glucose")
abline(v = mean(cohort.dt[["glucose"]], na.rm = TRUE), col = "red")
legend(x = "topright", lty = 1, col = "red", legend = "Mean", cex = 0.7,
       bg = "white")
```



Lastly, we analyse glucose, which has 44 missing observations. There are a few outliers which increase the mean over the dataset. This leads to the mean being larger than the median. However, we decide to proceed with mean imputation for glucose, as the values inputted are somehow close to the mean, but also reflect that some missing observations could be larger than the median. This decision is supported by the fact that only 44 observations are missing out of the 400 contained in the dataset, so no drastic difference in the final conclusions should arise.

Categorical variables

```
cat("There are", sum(is.na(cohort.dt[["diabetes"]])),
    "NA values for diabetes \nSimplified distribution tables for diabetes:")
```

```
There are 2 NA values for diabetes
Simplified distribution tables for diabetes:
```

```
diab.t <- table(cohort.dt$diabetes)
diabp.t <- round(prop.table(table(cohort.dt$diabetes)),3)

rownames(diab.t) <- c("No diabetes", "Diabetes")
kable(diab.t, format = "latex",
```

Table 6: Diabetes distribution (with NAs)

Diabetes status	Number of patients
No diabetes	263
Diabetes	135

Table 7: Diabetes distribution (percentage, with NAs)

Diabetes status	Frequency
No diabetes	0.661
Diabetes	0.339

```
      caption = "Diabetes distribution (with NAs)",
      align = "c",
      col.names = c('Diabetes status', 'Number of patients') )
```

```
rownames(diabp.t) <- c("No diabetes", "Diabetes")
kable(diabp.t, format = "latex",
      caption = "Diabetes distribution (percentage, with NAs)",
      align = "c",
      col.names = c('Diabetes status', 'Frequency'))
```

```
cat("\nSimplified distribution tables for diabetes after mode inputting:")
```

```
Simplified distribution tables for diabetes after mode inputting:
```

```
diab.t2 <- table(cohort.dt.copy$diabetes)
diabp.t2 <- round(prop.table(table(cohort.dt.copy$diabetes)),3)

rownames(diab.t2) <- c("No diabetes", "Diabetes")
kable(diab.t2, format = "latex",
      caption = "Diabetes distribution (mode imputation)",
      align = "c",
      col.names = c('Diabetes status', 'Number of patients'))
```

```
rownames(diabp.t2) <- c("No diabetes", "Diabetes")
kable(diabp.t2, format = "latex",
      caption = "Diabetes distribution (percentage, after mode imputation)",
```

Table 8: Diabetes distribution (mode imputation)

Diabetes status	Number of patients
No diabetes	265
Diabetes	135

Table 9: Diabetes distribution (percentage, after mode imputation)

Diabetes status	Frequency
No diabetes	0.662
Diabetes	0.338

```
      align = "c",
      col.names = c('Diabetes status', 'Frequency'))
```

```
##################################

cat("\n\nThere are", sum(is.na(cohort.dt[["albumin"]])),
    "NA values for albumin. \nSimplified distribution tables for albumin:")
```

```
There are 46 NA values for albumin.
Simplified distribution tables for albumin:
```

```
albu.t <- table(cohort.dt$albumin)
albup.t <- round(prop.table(table(cohort.dt$albumin)),3)


rownames(albu.t) <- c("Normo", "Micro", "Macro")
kable(albu.t, format = "latex",
      caption = "Albumin distribution (with NAs)",
      align = "c",
      col.names = c('Type', 'Number of patients'))
```

```
rownames(albup.t) <- c("Normo", "Micro", "Macro")
kable(albup.t, format = "latex",
      caption = "Albumin distribution (percentage, with NAs)",
      align = "c",
      col.names = c('Type', 'Frequency'))
```

Table 10: Albumin distribution (with NAs)

Type	Number of patients
Normo	199
Micro	130
Macro	25

Table 11: Albumin distribution (percentage, with NAs)

Type	Frequency
Normo	0.562
Micro	0.367
Macro	0.071

```
cat("\nSimplified distribution tables for albumin after mode inputting:")
```

```
Simplified distribution tables for albumin after mode inputting:
```

```
albu.t2 <- table(cohort.dt.copy$albumin)
albup.t2 <- round(prop.table(table(cohort.dt.copy$albumin)),3)


rownames(albu.t2) <- c("Normo", "Micro", "Macro")
kable(albu.t2, format = "latex",
      caption = "Albumin distribution (after mode imputation)",
      align = "c",
      col.names = c('Type', 'Number of patients'))
```

```
rownames(albup.t2) <- c("Normo", "Micro", "Macro")
kable(albup.t2, format = "latex",
      caption = "Albumin distribution (percentage, after mode imputation)",
      align = "c",
```

Table 12: Albumin distribution (after mode imputation)

Type	Number of patients
Normo	245
Micro	130
Macro	25

Table 13: Albumin distribution (percentage, after mode imputation)

Type	Frequency
Normo	0.613
Micro	0.325
Macro	0.062

```
        col.names = c('Type', 'Frequency'))
```

Above can be observed the distributions for categorical variables before and after mode inputting. In the case of diabetes, the distribution does not change much since there are only 2 missing observations. As diabetes can have a substantial effect in many parameters associated with health, and the number of missing values is small, we have decided that it is better to not input the most prominent value.

The data accounting for albuminuric status has 46 missing values. It can be seen that category 1 ("normo") dominates over "micro" and "macro". Thus, if feels safe to add the mode value and therefore not miss such a high number of data points that can be used for further study of the dataset.

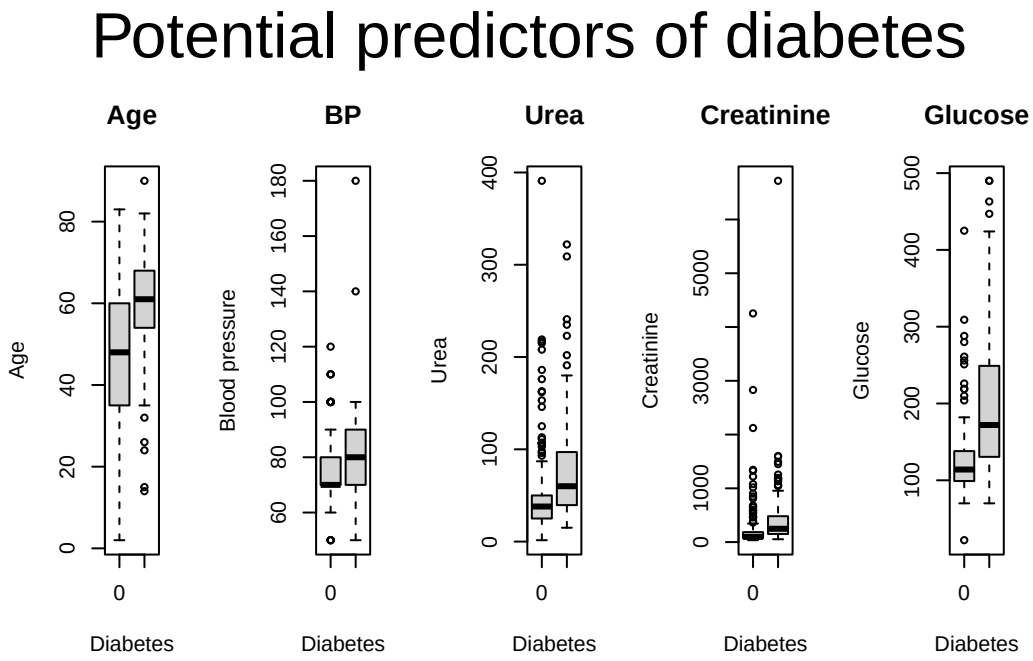
```
# Apply the mean/mode imputation as above
cohort.dt$age <- if_else(is.na(cohort.dt$age),
                                as.numeric(format(Sys.Date(), "%Y")) -
                            cohort.dt$yob,
                                      cohort.dt$age )
cohort.dt <- impute.to.mean("bp")
cohort.dt <- impute.to.mean("urea")
cohort.dt <- impute.to.mean("glucose")
cohort.dt <- impute.to.mode("albumin")
```

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.

```
# Set layout
par(mfrow=c(1,5),oma=c(0,0,2,0))
# Plot boxplots
```

```r
#text(0.5,0.5,"Potential predictors of diabetes",cex=2,font=2)
boxplot(cohort.dt$age ~ cohort.dt$diabetes, xlab = "Diabetes", ylab = "Age",
        main = "Age")
boxplot(cohort.dt$bp ~ cohort.dt$diabetes,
        xlab = "Diabetes", ylab = "Blood pressure", main = "BP")
boxplot(cohort.dt$urea ~ cohort.dt$diabetes, xlab = "Diabetes", ylab = "Urea",
        main = "Urea")
boxplot(cohort.dt$creatinine ~ cohort.dt$diabetes,
        xlab = "Diabetes", ylab = "Creatinine", main = "Creatinine")
boxplot(cohort.dt$glucose ~ cohort.dt$diabetes,
        xlab = "Diabetes", ylab = "Glucose", main = "Glucose")
mtext("Potential predictors of diabetes", line=0, side=3, outer=TRUE, cex=2)
```




```r
tab3c <- table(cohort.dt$albumin, cohort.dt$diabetes,
               dnn = c("Albuminuric status","Diabetes status"))

rownames(tab3c) <- c("Normo", "Micro", "Macro")
colnames(tab3c) <- c("No diabetes", "Diabetes")
kable(tab3c, format = "latex",
      caption = "Albumin distribution vs diabetes",
      align = "c")
```

Table 14: Albumin distribution vs diabetes

	No diabetes	Diabetes
Normo	192	53
Micro	61	69
Macro	12	13

From the boxplots above, the potentially most useful predictors seem to be glucose, urea and creatinine. The reasoning behind this is that the interquartile region for these variables barely overlap between the diabetes and not-diabetes cases.

It is interesting to study the table that displays the albuminuric status with regards to diabetes. About 80% of the patients classified with "normo" albuminuric status do not have diabetes, while this proportion grows to about 50% for both "micro" and "macro". Thus, we can conclude that having "normo" albuminuric status is significant when predicting diabetes.

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
# Subset required features
reg3d.dt <- cohort.dt[, c("diabetes", "creatinine", "glucose")]

# Define model
reg3d <- glm(diabetes ~ glucose + creatinine, data = reg3d.dt,
             family=binomial(link = "logit"))

summary(reg3d)
```

```
Call:
glm(formula = diabetes ~ glucose + creatinine, family = binomial(link = "logit"),
    data = reg3d.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9898  -0.7127  -0.5478   0.6726   2.2098
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.7377765  0.3802458  -9.830  < 2e-16 ***
glucose      0.0186004  0.0024630   7.552 4.29e-14 ***
creatinine   0.0010692  0.0003828   2.793  0.00522 **
---
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: 390.44  on 397  degrees of freedom
AIC: 396.44

Number of Fisher Scoring iterations: 5
```

For a model with two predictors, we have decided to use glucose and creatinine levels in the bloodstream, as they seem to be the variables that lead to a larger separation between diabetic and non-diabetic patients. We fitted a Generalised Linear Model (GLM), in particular a logistic regression model. The reason for choosing a logistic model is that diabetic condition is measured as a binary variable in our dataset, which classifies patients that are diabetic (cases) to be 1, and people who are not diabetic (controls) as 0. With this, it is assumed that cases and controls are drawn from a binomial distribution, and the logit link function is the standard for the binomial family.

As this model can be used in a clinical, real-life situation, we will not standardise the values we pass to the model, as the doctor/clinician will not have access to standard deviations. The results of the model tell us that both glucose and creatinine are positively associated to diabetes: larger levels of glucose and creatinine in the bloodstream increase the likelihood of a patient being diabetic. The estimated coefficients (glucose ≈ 0.0186 and creatinine ≈ 0.001069) represent how much the log-odds increase with each unit increase of the predictor. Note that, if the log-odds is larger than 0, the estimated probability of the patient being diabetic will be larger than 0.5, so the prediction will be that this patient is diabetic. Similarly, if the log-odds are negative, the patient will be predicted to not be diabetic. The estimated parameters are really significant for the dataset we passed on to the model.

The take-home message is that, the higher the glucose and creatinine levels in blood are, the more likely a person is to be diagnosed with diabetes.

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.

```r
# Subset required features
reg4a.dt <- cohort.dt[, c("diabetes", "creatinine", "glucose", "urea")]


# Define model
reg4a <- glm(diabetes ~ glucose + creatinine + urea, data = reg4a.dt,
             family=binomial(link = "logit"))

summary(reg4a)
```

```
Call:
glm(formula = diabetes ~ glucose + creatinine + urea, family = binomial(link = "logit"),
    data = reg4a.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0577  -0.6742  -0.5048   0.5706   2.2918

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.270e+00  4.231e-01 -10.093  < 2e-16 ***
glucose      1.854e-02  2.474e-03   7.492 6.78e-14 ***
creatinine   9.596e-05  3.006e-04   0.319     0.75
urea         1.353e-02  3.431e-03   3.944 8.01e-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: 374.22  on 396  degrees of freedom
AIC: 382.22

Number of Fisher Scoring iterations: 5
```

```r
# Likelihood-ratio test
pval <- pchisq(reg3d$deviance - reg4a$deviance, df=1, lower.tail=FALSE)
signif(pval, 2)
```

```
[1] 5.6e-05
```

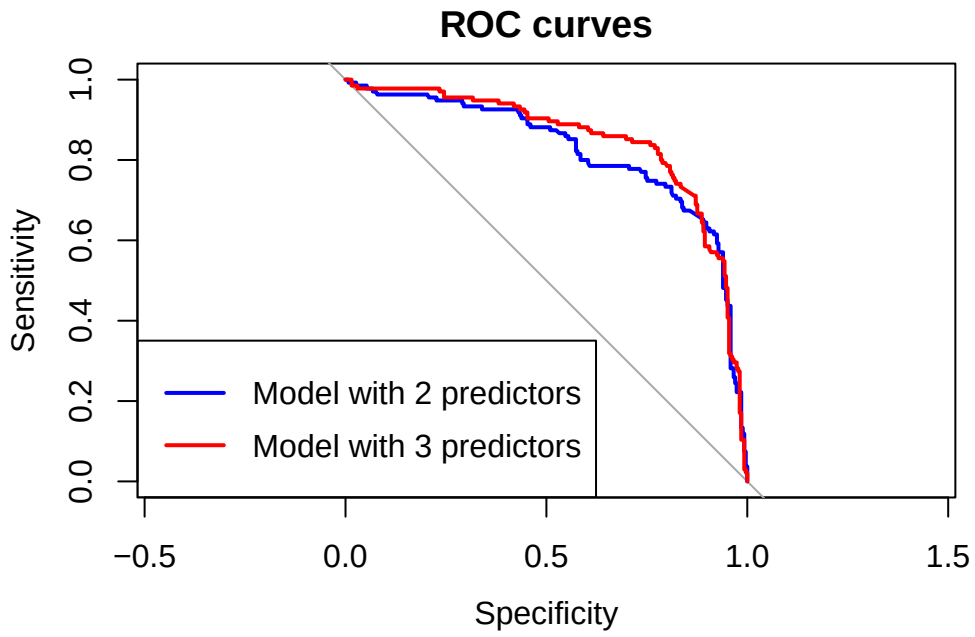
```r
# Same analysis, through a pre-built in function in R package "stats"
anova(reg3d, reg4a, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: diabetes ~ glucose + creatinine
Model 2: diabetes ~ glucose + creatinine + urea
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1       397     390.44
2       396     374.22  1   16.221 5.636e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case we run a similar model to the above, but adding a third predictor: urea levels in blood. It can be seen from the output above that this new model has a lower (residual) deviance that the one in exercise 3.d. After performing the likelihood ratio test, we obtain a p-value of approximately 5.6e-05. This p-value implies that the two models' deviances are statistically significantly different, so we have a reasonable argument to use the last, more complex model to predict diabetic status. In other words, the more complex model has a higher predictive power for diabetes.

```r
# Find predictions of the model
pred.3d <- predict(reg3d, type = "response")
pred.4a <- predict(reg4a, type = "response")

```

```r
# ROC curve, report AUC
suppressMessages(invisible({

roc.3d <- roc(reg3d.dt$diabetes,
              pred.3d, plot = TRUE,
              xlim = c(0,1),
              silent = TRUE,
              col = "blue",
              main = "ROC curves")
roc.4a <- roc(reg4a.dt$diabetes,
              pred.4a,
              plot = TRUE,
              silent = TRUE,
              col = "red",
              add = TRUE)

legend("bottomleft",
       legend = c("Model with 2 predictors", "Model with 3 predictors"),
       col = c("blue", "red"), lwd = 2)
}))
```



```r
cat("The AUC for the model with two predictors is", roc.3d$auc, "\nThe AUC for the model w
```

```
The AUC for the model with two predictors is 0.8206289
The AUC for the model with three predictors is 0.8483997
```

Above we see the plots of the ROC curves for both models developed, and their associated Area Under the Curve (AUC). The ROC curve is a representation of how well the models are able to predict that a patient has diabetes when it actually has diabetes, versus how well predicts a patient will not have diabetes when indeed the person does not have diabetes. Ideally, we would like these two measures to be 100%, that would be represented by both curves reaching the point in the top right corner (1,1).

A good statistic to measure the ROC curves meaning is the AUC. Ideally, we would want AUC to be 1, but in practice the AUC will almost always be smaller. For our models, we see that the AUCs are ≈ 0.82 (simpler model) and ≈ 0.85 (complex model). The AUC gives an overall view of the ability of a model to distinguish between classes, in our case diabetic and non-diabetic patients. As both models have a high AUC, close to each other, it means that they are both relatively well capable of predicting patients that suffer diabetes.

With the results of ROC and AUC, it can be argued that the simpler model performs almost as good as the complex model, and the fact that it takes less parameters makes it a better option when deciding which one to use. For similar predictive power, the simpler model may be more understandable and explainable. The statistically significant difference in the deviances may just be a consequence of using a large dataset passed on to the models.

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
# Define number of folds
num.folds <- 10

# Create folds
folds <- createFolds(reg3d.dt$diabetes, k = num.folds)

# Create some empty arrays to store values
regr.cv <- NULL
pred.cv <- NULL
roc.cv <- NULL
auc.cv <- NULL

```

```r
# For each fold, train and test models, store required values
for(f in 1:num.folds) {
  train.idx <- setdiff(1:nrow(reg3d.dt), folds[[f]])
  test.idx <- folds[[f]]

  regr.cv[[f]] <- glm(diabetes ~ glucose + creatinine, data = reg3d.dt,
                      subset = train.idx,
                      family=binomial(link = "logit"))
  pred.cv[[f]] <- predict(regr.cv[[f]],
                          type = "response",
                          newdata = reg3d.dt[folds[[f]]])
  suppressMessages(invisible({
  roc.cv[[f]] <- roc(reg3d.dt$diabetes[folds[[f]]], pred.cv[[f]])
  }))
}

# Take AUC for each model
for (i in 1:num.folds){
  auc.cv[i] <- roc.cv[[i]]$auc
}

# Calculate mean AUC
mean.auc <- mean(auc.cv)

cat("The mean cross-validated AUC across the 10 folds is", signif(mean.auc, 3))
```

```
The mean cross-validated AUC across the 10 folds is 0.824
```