Compulsory exercise: Team SuperGreat

MA8701 Advanced Statistical Learning V2023

Nora Aasen, Elias Angelsen, Jonas Nordstrom

15 mars, 2023

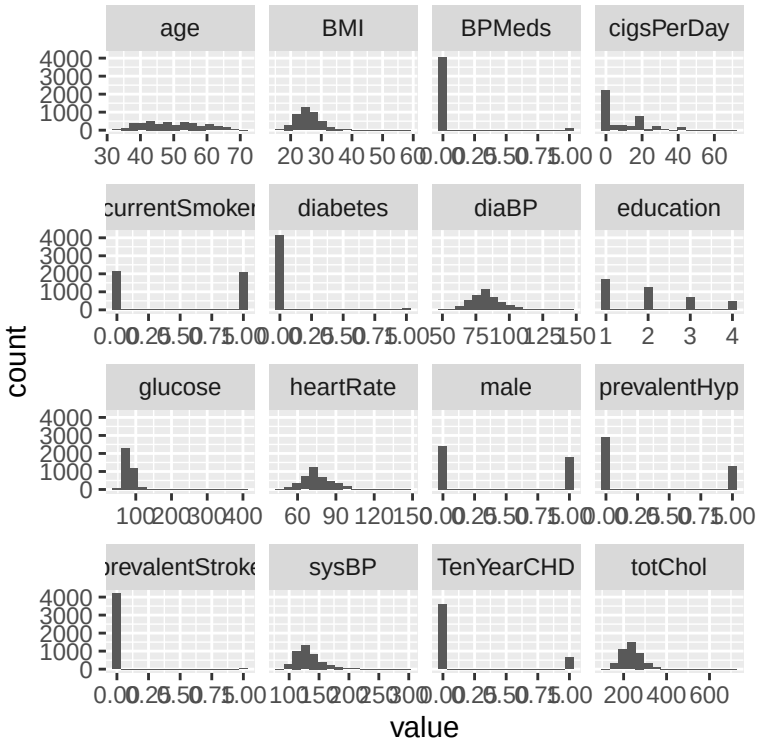
Introduction

In this project we have studied the Framingham Coronary Heart Disease Dataset. This dataset contains patient information for inhabitants in Framingham, Massachusetts, and is typically used to predict the chance of getting coronary heart disease (CHD) within the next 10 years. For this project, however, we intend to use lasso to find the most important risk factors. A big part of the task is to handle missing data. We will do single regression imputation manually and through the `mice` package, and investigate a bit what the Lasso is doing on the imputed data sets, compared to the complete case.

Exploratory Analysis

We start by examining the dataset.

```
# Load and look at data
data <- read.csv("C:\\Users\\noraa\\Student\\5thYear\\MA8701\\data_analysis_proj\\Heart_disease_analysis

data_dim = dim(data)
pos_response = sum(data$TenYearCHD==1)

# We visualize the data
ggplot(gather(data), aes(value)) +
    geom_histogram(bins = 16) +
    facet_wrap(~key, scales = 'free_x')
```

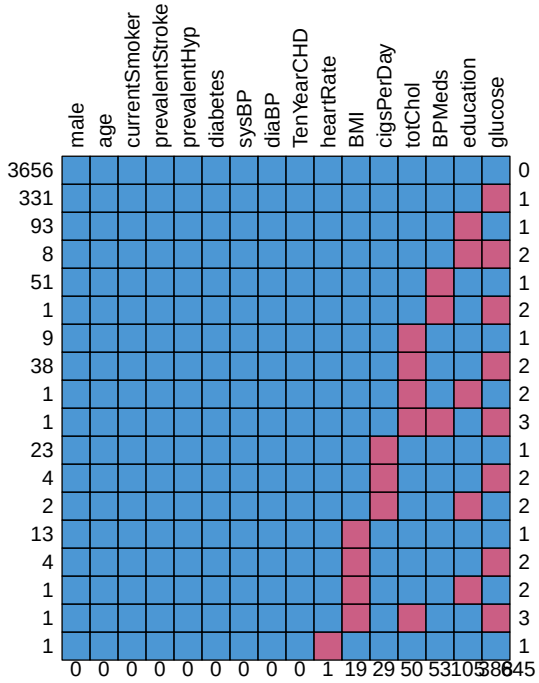



```
# Code education as a factor variable instead of 1-2-3-4.
data$education = factor(data$education, labels = c("none","hs","college","post-grad"))
```

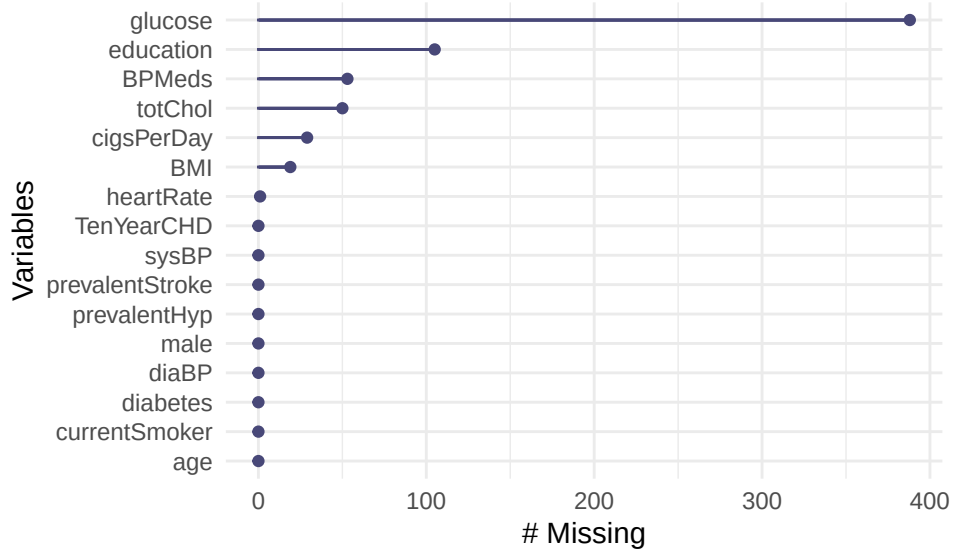
This data set contains 4238 observations, 15 covariates and a binary response variable `TenYearCHD`. We will try to fit a logistic regression model. The response variable has 644 observations that are 1, which equals about 15.2% of the total observations. Most of our covariates are either binary, or numeric. However, we notice that the variable education is most likely a categorical covariate. We could not find any further elaboration for which four categories the numbers represent, so based on the frequency of each value and qualified guessing, we changed it to a factor variable and defined the four categories as none, hs, college, post-grad.

The next thing we looked at was the number of missing data in our data set.

```
# Look at the missing data
md_mat = md.pattern(data, rotate.names = T, plot = T)
```




```
gg_miss_var(data)
```



As we can see there are six covariates that has missing data: `glucose`, `education`, `BPMeds`, `totChol`, `cigsPerDay`, and `BMI`. We cannot use the rows that contain missing values as is. The easiest solution is to remove all rows that contains `NA`'s. This is the *complete case* solution. We split the data into training and test sets, as well as copying the complete case data set for later solutions.

```
# Split into training and test first
set.seed(8701)
```

```r
tr = 7/10 # train ratio
r = dim(data)[1]
size = round(r*tr)
train = sample(1:r,size = size)

d.train = data[train,]
d.test = data[-train,]

# Make a dataset containing only the complete cases
d.complete <- data[complete.cases(data), ]
d.train.complete <- d.train[complete.cases(d.train), ]
d.test.complete <- d.test[complete.cases(d.test), ]

pos_response_c = sum(data[complete.cases(data),]$TenYearCHD==1)
```

The complete data set contains 3656 observations and the response variable has 557 observations that are 1, which equals about 15.2% of the total observations. As we can see, the proportion of positive observations in the response is the same, which is a good indicator that our data is missing at random (MAR), as we will discuss later.

Missing Data

We start by recalling that there are several types of mechanisms for missing data. Let $Z = (X, y)$ denote the full collection of covariates and responses, respectively, and we let a subscript mis/obs indicate whether we are restricting Z (or X) to the missing or observed parts, respectively. We may form an indicator (0-1) matrix R indicating missing (0) and observed (1) covariates. Assume ψ is short for the parameters in the distribution of R.

The missing data may be characterized by the conditioning in the distribution of R. We define the data to be:

- missing completely at random (MCAR) if $P(R|Z, \psi) = P(R|\psi)$,
- missing at random (MAR) if $P(R|Z, \psi) = P(R|Z_{obs}, \psi)$,
- missing not at random (MNAR) if $P(R|Z, \psi) = P(R|Z, \psi)$ (i.e. we don't have MCAR or MAR).

By exploring the missing pattern of for example the variable `cigsPerDay`, we see that our missing mechanism is not MCAR. No non-smoker has failed to answer the question ''How may cigarettes do you smoke a day?'', which is a question only aimed at smokers. The simple explanation may be that the survey automatically fills in 0 for `cigsPerDay` if you claim to be a non-smoker. In more mathematical terms, the missingness of `cigsPerDay` depends on the observed answer to''Do you smoke?" (found in variable `currentSmoker`), indicating that we do not work with MCAR data. Luckily, most methods are applicable if our missingness is at least MAR.

We will assume that the missing mechanism is MAR for all our missing observations, as there is no clear reason to suspect it to be MNAR.

To treat the missing data, we will use single imputation, as multiple imputation may cause difficulties with the resulting inference, as Rubin's rules needs to be combined with the Lasso, bootstrap and concluding inference. Multiple imputation was therefore not considered because it is beyond the scope of this project.

To perform single imputation we will use regression imputation, where we adapt our regression technique depending on the type of variable imputed. For continuous variables, we use a linear regression model, for

binary variables, we use logistic regression, and for the variable ''education", which is a four-class variable, we have utilized kNN for multiclass imputation. This is implemented manually, but this could have been done using the `MICE` package and the function `mice`.

To avoid encountering observations with more than one missing value, and hence problems with regressing, we remove all rows containing more than one `NA`.

Our data is split into training and test sets, with the test-to-training ratio being 3:7. In order to avoid data leakage in our imputation of the test set, we fit the imputation models on the training set. The main idea is that the test set should be viewed as several independent observations. Using the test set to impute itself will use information not present at the time of training and will yield unintended correlation. This is again done manually, but could also have been done using `mice.reuse`, as we will do later.

Note that we do not include the response (`TenYearCHD`) in the regression, and in order to avoid too much correlation between the imputed samples, we always base the regression models on the complete case data, instead of letting the imputed values for variable n regress to impute variable $n + 1$.

```
# Setting seed expressing our love for MA8701
set.seed(8701)

# Throwing out the samples with two or more NAs from the data
miss <- c()
for(i in 1:nrow(d.train)) {
  if(sum(is.na(d.train[i,])) > 1){
    miss <- append(miss,i)
    }
}

d.train.missing <- d.train[-miss,]


miss <- c()
for(i in 1:nrow(d.test)) {
  if(sum(is.na(d.test[i,])) > 1){
    miss <- append(miss,i)
    }
}

d.test.missing <- d.test[-miss,]
```

We further split the data into further training and test sets with and without the response `TenYearCHD`, following the training-test-ratio that we have previously set.

```
x.train_dat_m = subset(d.train.missing, select = -TenYearCHD)
y.train.miss = d.train.missing$TenYearCHD

x.test_dat_m = subset(d.test.missing, select = -TenYearCHD)
y.test.miss = d.test.missing$TenYearCHD
```

Using these data sets, we make regression models for each missing variable based on the data we have. These models automatically neglects NA's. For `glucose`, `cigsPerDay`, `BMI`, `totChol` and `heartRate`, we fit linear models, while for the binary variable `BPMeds`, a logistic model is fit. The multi-class variable `education` will be imputed using a kNN-model.

```r
# We fit linear models on the training set for glucose, cigsPerDay, BMI, totChol, heartRate.
fit_for_simp_glucose_m <- lm(glucose ~., data = x.train_dat_m)    # Linear model for glucose
fit_for_simp_cigs_m <- lm(cigsPerDay ~., data = x.train_dat_m)    # Linear model for cigsPerDay
fit_for_simp_BMI_m <- lm(BMI ~., data = x.train_dat_m)            # Linear model for BMI
fit_for_simp_totChol_m <- lm(totChol ~., data = x.train_dat_m)    # Linear model for totChol
fit_for_simp_heartRate_m <- lm(heartRate ~., data = x.train_dat_m)# Linear model for heartRate

# We fit a logistic model on the training set for BPMeds, being a binary variable.
fit_for_simp_BPMeds_m <- glm(BPMeds ~., data = x.train_dat_m, family = "binomial") #Logistic model for

# We fit a KNN model on the training set for education, being a multi-class variable.
fit_for_simp_edu_m <- gknn(education ~., data = x.train_dat_m) # kNN model for education
```

To predict the missing values for each variable, we pick out those samples with missing values of each variable from the training and test set. Note that we are imputing for both the training and test set, even though the sampled (therefore "random") split into training and test set may have sorted all NA's of a variable (e.g. heartRate) into a single set (i.e. into either training or test set). This is not a problem, as we are only picking out the incomplete cases. Therefore, predicting and filling in the missing values in the complete part is a vacuous procedure, not yielding any problems.

```r
# Pick out those variables with missing glucose from the training and test set.
train_data_to_pred_gluc_m = ic(x.train_dat_m)
train_data_to_pred_gluc_m = train_data_to_pred_gluc_m[ici(train_data_to_pred_gluc_m$glucose),]
test_data_to_pred_gluc_m = ic(x.test_dat_m)
test_data_to_pred_gluc_m = test_data_to_pred_gluc_m[ici(test_data_to_pred_gluc_m$glucose),]

# Pick out those variables with missing cigsPerDay from the training and test set.
train_data_to_pred_cigs_m = ic(x.train_dat_m)
train_data_to_pred_cigs_m = train_data_to_pred_cigs_m[ici(train_data_to_pred_cigs_m$cigsPerDay),]
test_data_to_pred_cigs_m = ic(x.test_dat_m)
test_data_to_pred_cigs_m = test_data_to_pred_cigs_m[ici(test_data_to_pred_cigs_m$cigsPerDay),]

# Pick out those variables with missing BMI from the training and test set.
train_data_to_pred_BMI_m = ic(x.train_dat_m)
train_data_to_pred_BMI_m = train_data_to_pred_BMI_m[ici(train_data_to_pred_BMI_m$BMI),]
test_data_to_pred_BMI_m = ic(x.test_dat_m)
test_data_to_pred_BMI_m = test_data_to_pred_BMI_m[ici(test_data_to_pred_BMI_m$BMI),]

# Pick out those variables with missing totChol from the training and test set.
train_data_to_pred_totChol_m = ic(x.train_dat_m)
train_data_to_pred_totChol_m = train_data_to_pred_totChol_m[ici(train_data_to_pred_totChol_m$totChol),]
test_data_to_pred_totChol_m = ic(x.test_dat_m)
test_data_to_pred_totChol_m = test_data_to_pred_totChol_m[ici(test_data_to_pred_totChol_m$totChol),]

# Pick out those variables with missing heartRate from the training and test set.
train_data_to_pred_heartRate_m = ic(x.train_dat_m)
train_data_to_pred_heartRate_m = train_data_to_pred_heartRate_m[ici(train_data_to_pred_heartRate_m$hear
test_data_to_pred_heartRate_m = ic(x.test_dat_m)
test_data_to_pred_heartRate_m = test_data_to_pred_heartRate_m[ici(test_data_to_pred_heartRate_m$heartRa

# Pick out those variables with missing BPMeds from the training and test set.
train_data_to_pred_BPMeds_m = ic(x.train_dat_m)
train_data_to_pred_BPMeds_m = train_data_to_pred_BPMeds_m[ici(train_data_to_pred_BPMeds_m$BPMeds),]
```

```
test_data_to_pred_BPMeds_m = ic(x.test_dat_m)
test_data_to_pred_BPMeds_m = test_data_to_pred_BPMeds_m[ici(test_data_to_pred_BPMeds_m$BPMeds),]

# Pick out those variables with missing education from the training and test set.
train_data_to_pred_edu_m = ic(x.train_dat_m)
train_data_to_pred_edu_m = train_data_to_pred_edu_m[ici(train_data_to_pred_edu_m$education),]
test_data_to_pred_edu_m = ic(x.test_dat_m)
test_data_to_pred_edu_m = test_data_to_pred_edu_m[ici(test_data_to_pred_edu_m$education),]
```

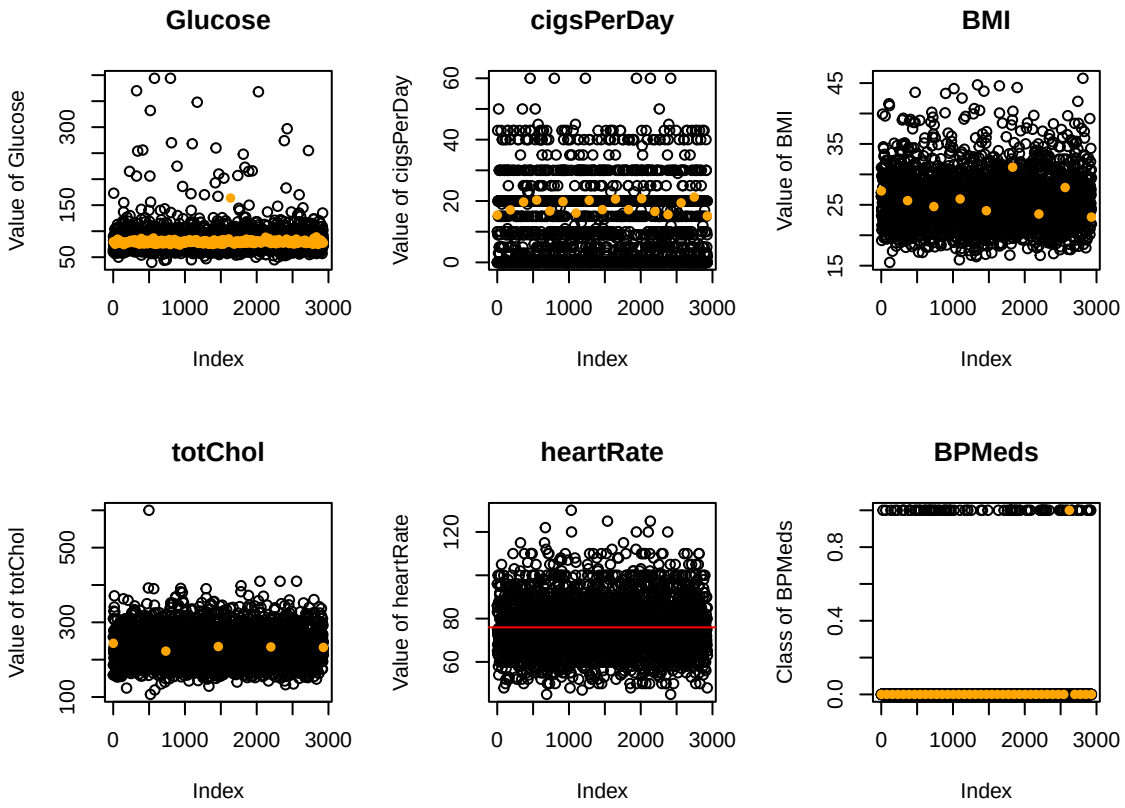
We then predict the missing values for imputation. For the logistic model fit to predict `BPMeds`, we assign a prediction class 1 or 0 depending on whether the predicted value is above or below 0.5.

```
set.seed(8701)
# Predicting the values to impute for glucose on the training and test data.
pred_glucose_train_m <- predict(fit_for_simp_glucose_m, newdata = train_data_to_pred_gluc_m)
pred_glucose_test_m <- predict(fit_for_simp_glucose_m, newdata = test_data_to_pred_gluc_m)

# Predicting the values to impute for cigsPerDay on the training and test data.
pred_cigs_train_m <- predict(fit_for_simp_cigs_m, newdata = train_data_to_pred_cigs_m)
pred_cigs_test_m <- predict(fit_for_simp_cigs_m, newdata = test_data_to_pred_cigs_m)

# Predicting the values to impute for BMI on the training and test data.
pred_BMI_train_m <- predict(fit_for_simp_BMI_m, newdata = train_data_to_pred_BMI_m)
pred_BMI_test_m <- predict(fit_for_simp_BMI_m, newdata = test_data_to_pred_BMI_m)

# Predicting the values to impute for totChol on the training and test data.
pred_totChol_train_m <- predict(fit_for_simp_totChol_m, newdata = train_data_to_pred_totChol_m)
pred_totChol_test_m <- predict(fit_for_simp_totChol_m, newdata = test_data_to_pred_totChol_m)

# Predicting the values to impute for heartRate on the training and test data.
pred_heartRate_train_m <- predict(fit_for_simp_heartRate_m, newdata = train_data_to_pred_heartRate_m)
pred_heartRate_test_m <- predict(fit_for_simp_heartRate_m, newdata = test_data_to_pred_heartRate_m)

# Predicting the classes to impute for BPMeds on the training and test data by first predicting numeric
pred_BPMeds_train_m_probs <- predict(fit_for_simp_BPMeds_m, newdata = train_data_to_pred_BPMeds_m, type
pred_BPMeds_test_m_probs <- predict(fit_for_simp_BPMeds_m, newdata = test_data_to_pred_BPMeds_m, type =
pred_BPMeds_train_m <- ifelse(pred_BPMeds_train_m_probs >= 0.5, 1, 0)
pred_BPMeds_test_m <- ifelse(pred_BPMeds_test_m_probs >= 0.5, 1, 0)

# Predicting the classes to impute for education on the training and test data.
pred_edu_train_m <- predict(fit_for_simp_edu_m, newdata = train_data_to_pred_edu_m, type = "class")
pred_edu_test_m <- predict(fit_for_simp_edu_m, newdata = test_data_to_pred_edu_m, type = "class")
```

To see how our predictions compare to the complete case, we plot (in different ways) the predicted values/classes. For illustrating different types of plots useful for this, we plot glucose twice, as both a transparent histogram and a point plot. This is only for illustrational purposes and therefore only done for the training data with its imputed values.

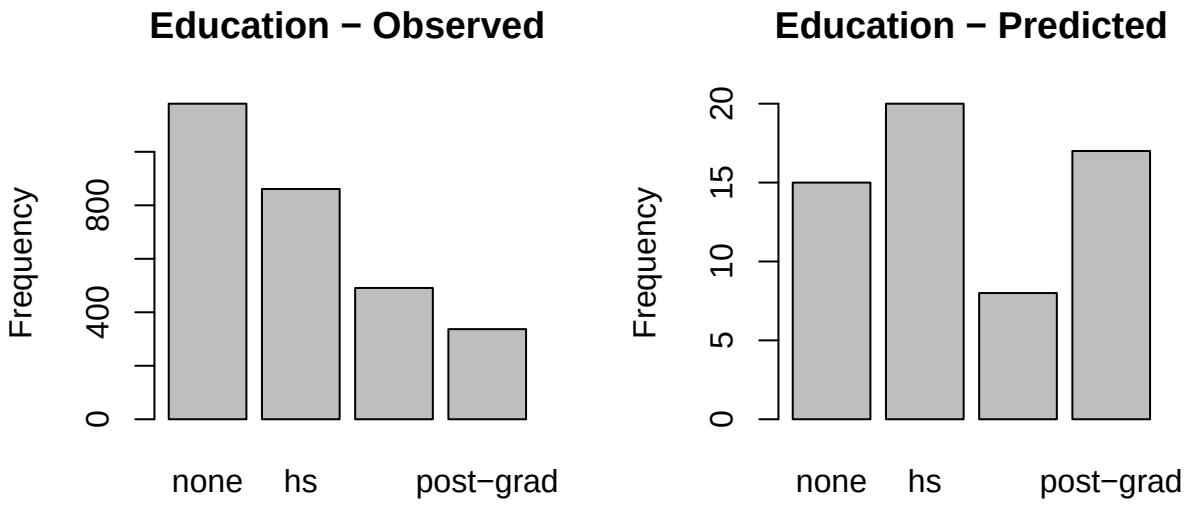
```
# ---- Make transparent colors for plotting ----
c1 <- rgb(173,216,230,max = 255, alpha = 80, names = "lt.blue")
c2 <- rgb(255,192,203, max = 255, alpha = 80, names = "lt.pink")

# ---- Make histogram objects for glucose ----
```

```r
# Histogram objects for glucose
hg_pred_gluc = hist(pred_glucose_train_m, plot = FALSE)
hg_old_gluc = hist(x.train_dat_m$glucose, plot = FALSE)


# ---- Make point plots over indexes for all except education ----

par(mfrow = c(2,3))

# Point plot for glucose
plot(x.train_dat_m$glucose, ylab = "Value of Glucose", main = "Glucose")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_glucose_train_m)),pred_glucose_train_m,
abline(h = mean(x.train_dat_m$glucose), col = "red")

# Point plot for cigsperday
plot(x.train_dat_m$cigs, ylab = "Value of cigsPerDay", main = "cigsPerDay")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_cigs_train_m)),pred_cigs_train_m, col =
abline(h = mean(x.train_dat_m$cigsPerDay), col = "red")

# Point plot for BMI
plot(x.train_dat_m$BMI, ylab = "Value of BMI", main = "BMI")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_BMI_train_m)),pred_BMI_train_m, col = "
abline(h = mean(x.train_dat_m$BMI), col = "red")

# Point plot for totChol
plot(x.train_dat_m$totChol, ylab = "Value of totChol", main = "totChol")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_totChol_train_m)),pred_totChol_train_m,
abline(h = mean(x.train_dat_m$totChol), col = "red")

# Point plot for heartRate
plot(x.train_dat_m$heartRate, ylab = "Value of heartRate", main = "heartRate")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_heartRate_train_m)),pred_heartRate_trai
abline(h = mean(x.train_dat_m$heartRate), col = "red")

# Point plot for BPMeds
plot(x.train_dat_m$BPMeds, ylab = "Class of BPMeds", main = "BPMeds")
points(seq(1,length(x.train_dat_m[,1]),length.out = length(pred_BPMeds_train_m)),pred_BPMeds_train_m, c
abline(h = mean(x.train_dat_m$BPMeds), col = "red")
```




```
# ---- Make histogram plots for glucose and education ----

par(mfrow = c(1,1))

# Histogram plots for glucose
plot(hg_old_gluc, col = c1, xlab = "Value of Glucose", main = "Glucose") # Plot 1st histogram using a t
plot(hg_pred_gluc, , col = c2, add = TRUE)
legend("topright", legend=c("Observed - blue", "Predictions - pink"), cex=0.6)
```


```
par(mfrow = c(1,2))

# Histograms for education
plot(x.train_dat_m$education, main = "Education - Observed", ylab = "Frequency")
plot(pred_edu_train_m, main = "Education - Predicted", ylab = "Frequency")
```



For most of the linear predictors, we are imputing using linear models. As we can see in these plots, we are close to the mean, but we are including more variance than what mean imputation would be able to. The binary variable `BPMeds` is often classified to zero, which is expected, as the variable explains whether

or not the patient was taking blood pressure medicine at the time of the survey. The `education` variable is predicted using kNN, and we predict surprisingly many with higher education. This can imply that the kNN-model we fit is not optimal for predicting `education`. Lastly, we update our data with the newly imputed values.

```
# ---- Making new data sets based on the old ones ----
x.train_imp = x.train_dat_m
x.test_imp = x.test_dat_m

# ---- We add the predictions to this data set. ----

# Adding glucose
x.train_imp[ici(x.train_imp$glucose),]$glucose <- pred_glucose_train_m
x.test_imp[ici(x.test_imp$glucose),]$glucose <- pred_glucose_test_m

# Adding cigsPerDay
x.train_imp[ici(x.train_imp$cigsPerDay),]$cigsPerDay <- pred_cigs_train_m
x.test_imp[ici(x.test_imp$cigsPerDay),]$cigsPerDay <- pred_cigs_test_m

# Adding BMI
x.train_imp[ici(x.train_imp$BMI),]$BMI <- pred_BMI_train_m
x.test_imp[ici(x.test_imp$BMI),]$BMI <- pred_BMI_test_m

# Adding totChol
x.train_imp[ici(x.train_imp$totChol),]$totChol <- pred_totChol_train_m
x.test_imp[ici(x.test_imp$totChol),]$totChol <- pred_totChol_test_m

# Adding heartRate
x.train_imp[ici(x.train_imp$heartRate),]$heartRate <- pred_heartRate_train_m
x.test_imp[ici(x.test_imp$heartRate),]$heartRate <- pred_heartRate_test_m

# Adding BPMeds
x.train_imp[ici(x.train_imp$BPMeds),]$BPMeds <- pred_BPMeds_train_m
x.test_imp[ici(x.test_imp$BPMeds),]$BPMeds <- pred_BPMeds_test_m

# Adding education
x.train_imp[ici(x.train_imp$education),]$education <- pred_edu_train_m
x.test_imp[ici(x.test_imp$education),]$education <- pred_edu_test_m


# Construct final imputed data sets

d.train.imp = x.train_imp
d.test.imp = x.test_imp

d.train.imp["TenYearCHD"] = y.train.miss
d.test.imp["TenYearCHD"] = y.test.miss
```

We have managed to impute the missing values, but we have already some thoughts on how this procedure could be improved.

First of all, we could have used more flexible models for imputation than linear imputation, and we could have fit several different models on the training set and done evaluation procedures within the training set (e.g. cross validated optimization of ROC-AUC) to pick the models before fixing a model to impute each variable.

Some variables, such as `cigsPerDay`, are in some sense discrete, although we have treated them as continuous (as it is possible to smoke e.g. $2, 73$ cigarettes per day). These could have been rounded off to integers, but we didn't see why this would be necessary. We did not include the response (`TenYearCHD`) in the regressions. It is not clear to us why it would be a better choice to include it, as we don't want the imputed data to be overfitted towards the response. More reading and testing would be needed to figure out the optimal solution, if there is any canonical choice, and it would be interesting to see how our results changed if the response were included.

For less code, the `MICE` package could have been used, referring to the function `mice` for imputation of the training set and `mice.reuse` to reuse the imputation models on the test set.

Model

In the model section we will consider the two data sets; the complete case and imputed case. Both data sets are further divided into a train and test set. Since we want to do Lasso, we must standardize the data. The problem with this is data leakage. If we want to standardize the test data, we should standardize it using the mean and the standard deviation of the training data. Most importantly, using the test data to scale the test data will introduce correlation between the independent observations of the test set. Since the scaling information from the test set is "not available" to us at the time of training, we cannot expect the coefficients in the Lasso to be appropriately scaled compared to the test data. We solve this by scaling the training data, and then using the attributes of the training data to scale the test data accordingly.

```
# Make the training data ready for lasso by scaling.

set.seed(8701)

# Scale complete data for training

x.train.complete = scale(model.matrix(TenYearCHD ~ . -1, data = d.train.complete, family = binomial()))
train.complete.mean = attr(x.train.complete, "scaled:center")
train.complete.sd = attr(x.train.complete, "scaled:scale")

y.train.complete = d.train.complete$TenYearCHD

# Imputed data training set scaling

x.train.imp = scale(model.matrix(TenYearCHD ~ . -1, data = d.train.imp, family = binomial()))
train.imp.mean = attr(x.train.imp, "scaled:center")
train.imp.sd = attr(x.train.imp, "scaled:scale")
y.train.imp = d.train.imp$TenYearCHD

# Same for imputed test data, but depending on training set attributes.
x.test.imp = scale(model.matrix(TenYearCHD ~ . -1, data = d.test.imp, family = binomial()), center = tra
```

Given the binary response it is natural to consider fitting a logistic regression model to our data. Although we intend to use lasso, it is nice to start by fitting a regular logistic regression model on the complete case data to get an indication of which covariates that are most present, and for later comparison. We obtain the regression coefficients, a confusion matrix, a ROC-curve and the ROC-AUC.

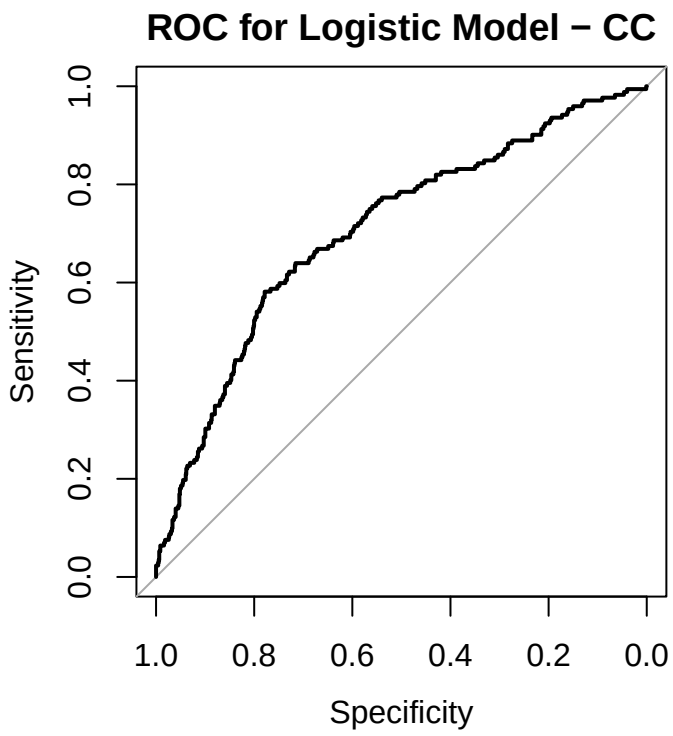
```
# Fit a logistic model
set.seed(8701)
mod0 <- glm(TenYearCHD ~ ., data = d.train.complete, family = binomial())
round(summary(mod0)$coefficients,7)
```

```
##                       Estimate Std. Error    z value  Pr(>|z|)
## (Intercept)        -8.1909517  0.8648439 -9.4710177 0.0000000
## male                0.6102028  0.1348701  4.5243748 0.0000061
## age                 0.0644352  0.0082290  7.8302194 0.0000000
## educationhs        -0.3167564  0.1532992 -2.0662624 0.0388037
## educationcollege   -0.2142688  0.1813577 -1.1814710 0.2374157
## educationpost-grad -0.1121211  0.1954944 -0.5735261 0.5662885
## currentSmoker      -0.0122323  0.1911538 -0.0639921 0.9489765
## cigsPerDay          0.0209125  0.0076803  2.7228564 0.0064720
## BPMeds              0.3232638  0.2692250  1.2007202 0.2298598
## prevalentStroke    -0.0717723  0.7060112 -0.1016589 0.9190275
## prevalentHyp        0.3578473  0.1679542  2.1306245 0.0331201
## diabetes           -0.1037130  0.3737342 -0.2775047 0.7813926
## totChol             0.0025062  0.0013865  1.8075291 0.0706798
## sysBP               0.0155640  0.0046170  3.3710343 0.0007489
## diaBP              -0.0051849  0.0078536 -0.6601879 0.5091332
## BMI                -0.0004882  0.0155704 -0.0313554 0.9749861
## heartRate          -0.0041223  0.0051729 -0.7968906 0.4255146
## glucose             0.0074510  0.0027470  2.7123748 0.0066803
```

```
mod0_preds <- predict(mod0, newdata = d.test.complete, type = "response")

predicted_value <- factor(round(mod0_preds))
expected_value <- factor(d.test.complete$TenYearCHD)

conf.mat.cc = confusionMatrix(data=predicted_value, reference = expected_value)$table
conf.mat.cc
```

```
##           Reference
## Prediction   0   1
##          0 917 161
##          1   9  11
```

```
roc_obj_cc <- roc(d.test.complete$TenYearCHD, mod0_preds, levels = c(0,1), direction = "<")
plot(roc_obj_cc, main = "ROC for Logistic Model - CC", cex = 0.5)
```




```
auc_cc = auc(roc_obj_cc)
auc_cc
```

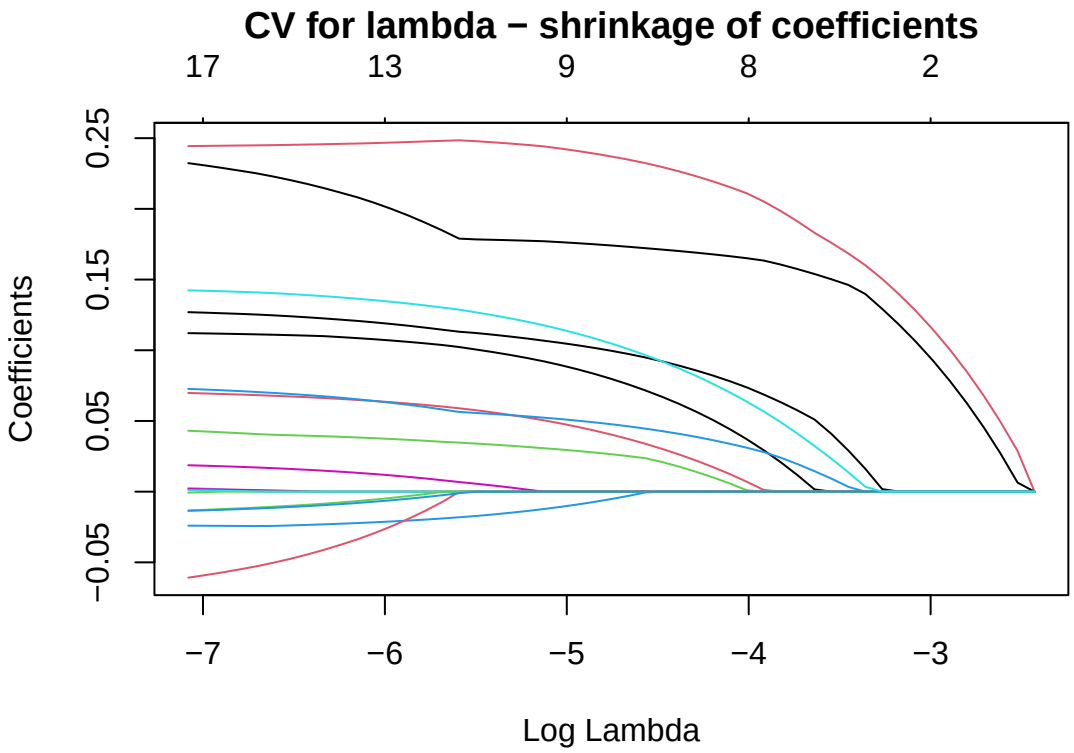
```
## Area under the curve: 0.7054
```

The logistic regression model chooses (Intercept), male, age, cigsPerDay, sysBP, glucose as the significant covariates, where the p-value cutoff is 0.01. It classifies very few positives correctly, which is very problematic if the model would be used to predict hearth disease.

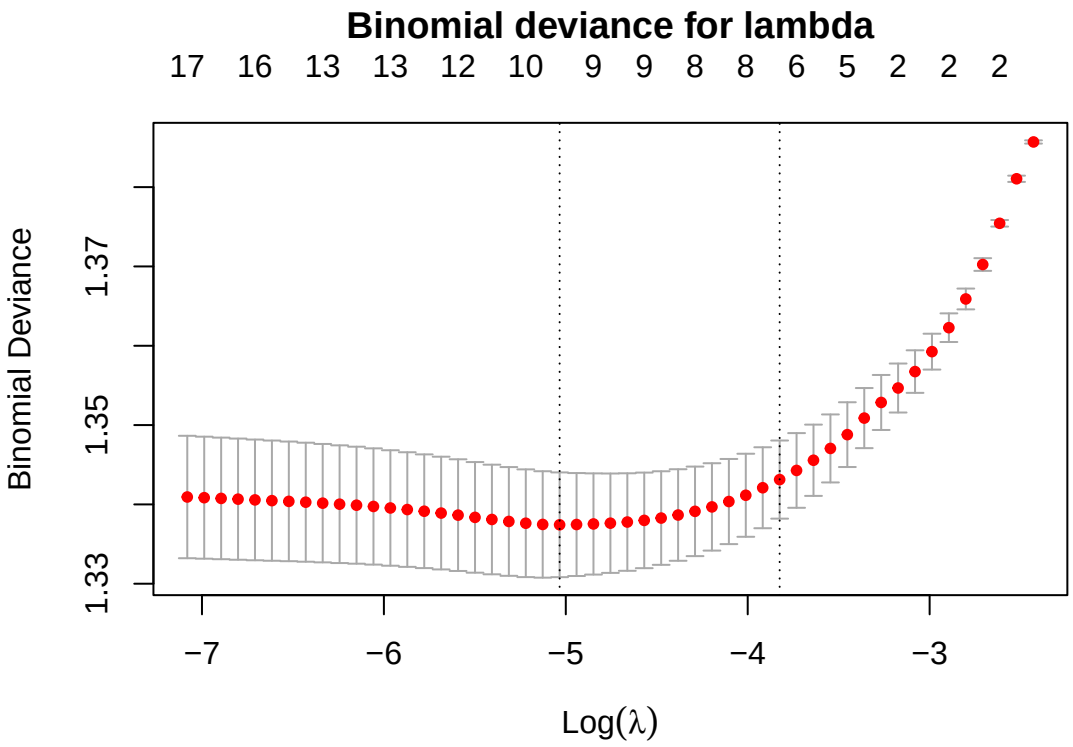
Lasso on Complete Case

We continue to do the Lasso on the complete case data. To do this, we use cross-validation to find λ_{min} and use the highest λ with deviance within one standard deviation of λ_{min}. We cross-validate for λ and plot the shrinkage and binomial deviance.

```
#, fig.height = 4.3 , fig.width=6
set.seed(8701)

# Use cross-validation to find lambda
cv.out = cv.glmnet(x.train.complete, y.train.complete, family = "binomial",intercept = F, standardize=T
plot(cv.out$glmnet.fit, "lambda", label=F, main = c("CV for lambda - shrinkage of coefficients",""))
```



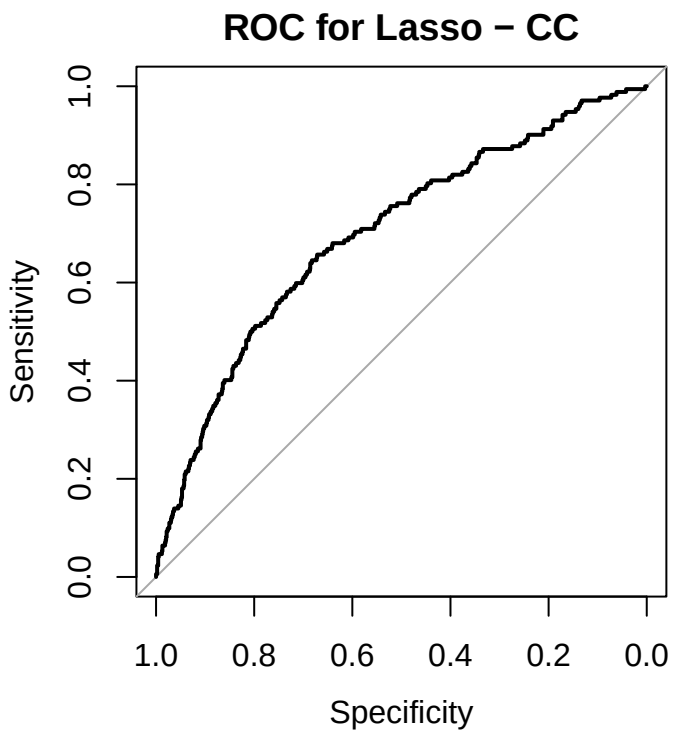

```
plot(cv.out, main = c("Binomial deviance for lambda",""))
```




```
# Fit the Lasso model

lasso_mod_cc = glmnet(x.train.complete, y.train.complete, family = "binomial",intercept = F, standardiz

lasso_coef_cc <- coef(lasso_mod_cc,s=cv.out$lambda.1se)
```

The confusion table, ROC and ROC-AUC is given below.

```
##            Reference
## Prediction   0   1
##          0 533  50
##          1 393 122
```



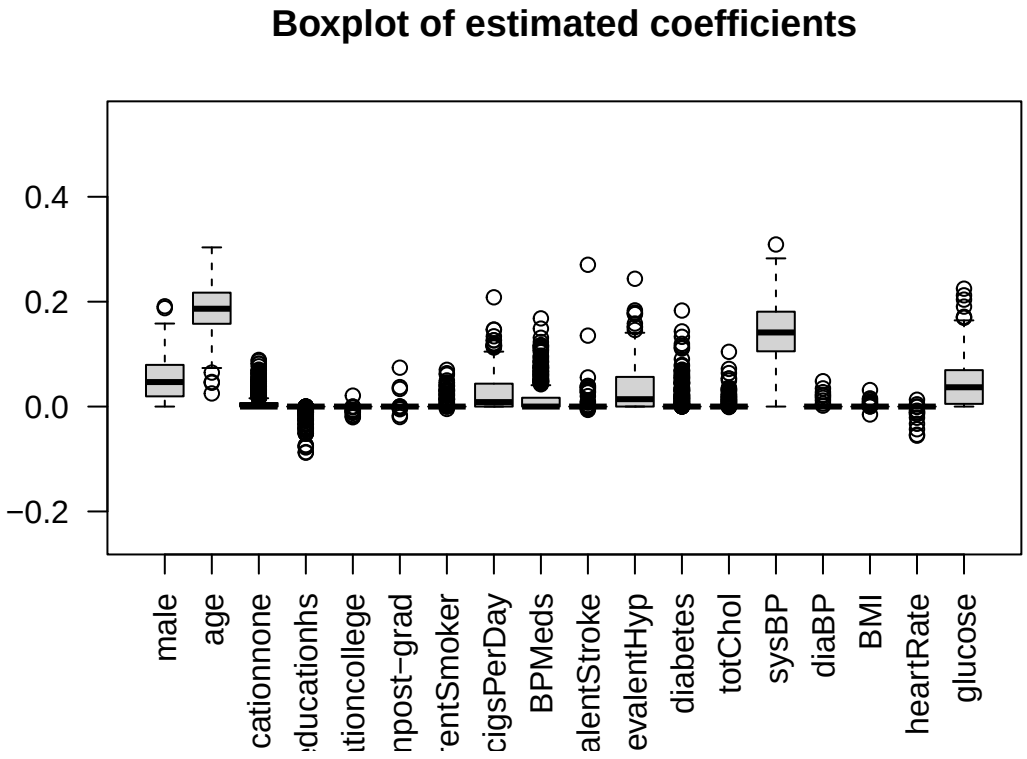

```
## Area under the curve: 0.6983
```

The Lasso on the complete case data chooses male, age, cigsPerDay, prevalentHyp, sysBP, glucose as the significant covariates. Initially, this is the same lasso model is much better at classifying positives than the full logistic model and has only slightly worse AUC, at 0.6982929 for the Lasso versus 0.705441 for the logistic model. We see that the Lasso includes `prevalentHyp`, while the logistic model finds it almost significant, with p-value 0.0331201. This agrees with the reasoning we made earlier, saying it could be an important parameter, but not as important as (either) `sysBP` or `diaBP`.

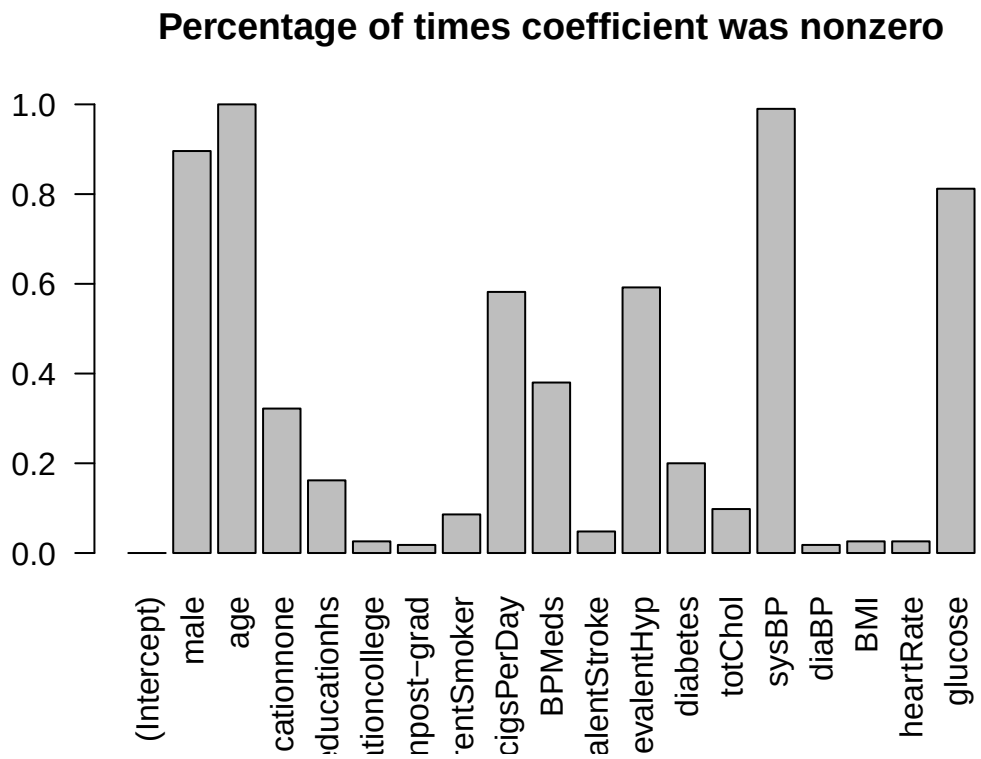
To obtain a better understanding of these coefficients, we bootstrap from the training data to fit Lasso models and store their coefficients.

```
set.seed(8701)

B = 500
boot_size_cc = dim(d.train.complete)[1]

B_coef_cc = matrix(NA, nrow = B, ncol = length(lasso_coef_cc[,1]))
for (i in 1:B){
  data_b = sample(1:boot_size_cc, size = boot_size_cc, replace = TRUE)
  x = x.train.complete[data_b,]
  y = y.train.complete[data_b]
  cv.out = cv.glmnet(x, y, family = "binomial", alpha = 1)
  lasso_mod = glmnet(x, y, family = "binomial", alpha = 1, intercept = F, lasso = cv.out$lambda.1se)

  B_coef_cc[i,] <- coef(lasso_mod,s=cv.out$lambda.1se)[,1]
}
```

```
colnames(B_coef_cc) = names(coef(lasso_mod,s=cv.out$lambda.1se)[,1])
boxplot.matrix(B_coef_cc[,-1], ylim = c(-0.25,0.55), las = 2, main = "Boxplot of estimated coefficients
```




```
B_coef_count_cc = ifelse(B_coef_cc == 0,0,1)
barplot(apply(B_coef_count_cc, 2, sum)/B, las = 2, main = "Percentage of times coefficient was nonzero"
```




```
names_lasso_cc_70 <- names(apply(B_coef_count_cc, 2, sum)/B)[apply(B_coef_count_cc, 2, sum)/B > 0.7]
names_lasso_cc_50 <- names(apply(B_coef_count_cc, 2, sum)/B)[apply(B_coef_count_cc, 2, sum)/B > 0.5]
```

We ran the bootstrap using 500 iterations. The variables that have nonzero coefficients in the Lasso models at least 70% of the time, are male, age, sysBP, glucose. Similarly, by those to those who are included at least 50% of the time, we obtain male, age, cigsPerDay, prevalentHyp, sysBP, glucose. This is indeed similar to the ones we picked out earlier.

Lasso on Imputed Data

We now do the same thing, just using the imputed data instead of the complete case. Even though the imputed data includes more samples, the data quality is going down when we impute.

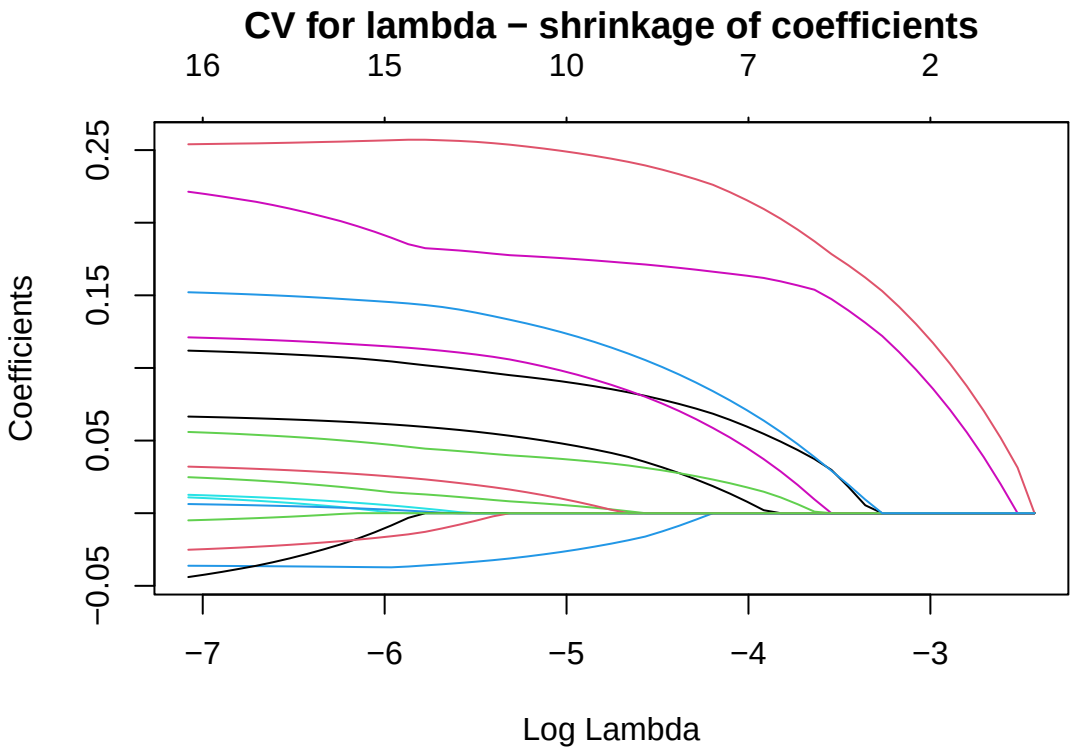
To get hands-on experience with the `MICE` package, we also construct an imputed data set using `mice` and `mice.reuse`, for comparison. `mice` can be used on the entire training data, without needing to remove those samples with two or more covariates missing. The imputation model from the training set is emplyed to impute the test set as well, to avoid data leakage.

```
set.seed(8701)

# Important that we don't use the response as predictor when imputing
y.train <- d.train$TenYearCHD
y.test <- d.test$TenYearCHD

d.train["TenYearCHD"] <- NULL
d.test["TenYearCHD"] <- NULL
```

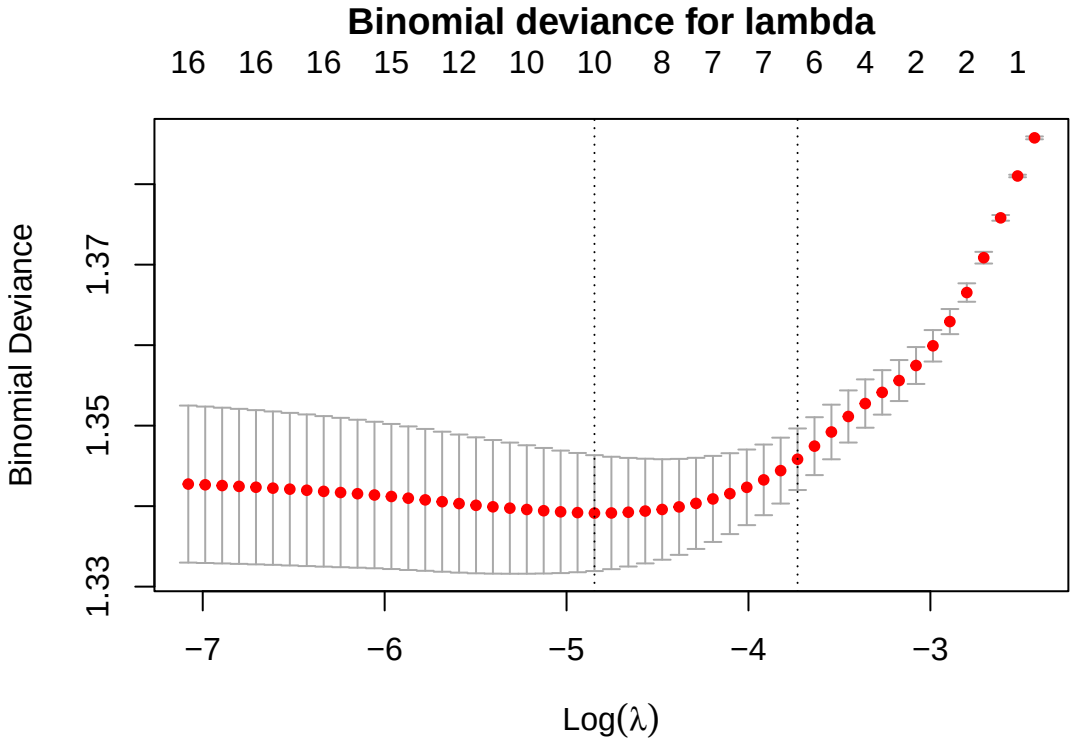
```
imp_train = mice(d.train, m=1, printFlag = F)
d.train.imp.mice = complete(imp_train)

# Use the imputation model trained on the training set on the test set, to ensure no data leakage.
imp_test = mice.reuse(imp_train, d.test, printFlag = F)
d.test.imp.mice = imp_test$"1"

# We scale the model matrix as before.

x.train.imp.mice = scale(model.matrix(~ . -1, data = d.train.imp.mice, family = binomial()))
train.imp.mean.mice = attr(x.train.imp.mice, "scaled:center")
train.imp.sd.mice = attr(x.train.imp.mice, "scaled:scale")
y.train.imp.mice = d.train.imp$TenYearCHD

# Same for test data, but depending on attributes from the training data in order to avoid data leakage
x.test.imp.mice = scale(model.matrix(~ . -1, data = d.test.imp.mice, family = binomial()), center = tra
```

First we try the Lasso on the imputed dataset where we imputed with our manual technique. We show the shrinkage and the binomial deviance over λ, which is a part of the cross-validation

```
# We need to standardize the matrix so that we can drop intercepts.
cv.out = cv.glmnet(x.train.imp, y.train.imp, family = "binomial", intercept = F, standardize=TRUE, alpha
plot(cv.out$glmnet.fit, "lambda", label=F, main = c("CV for lambda - shrinkage of coefficients",""))
```



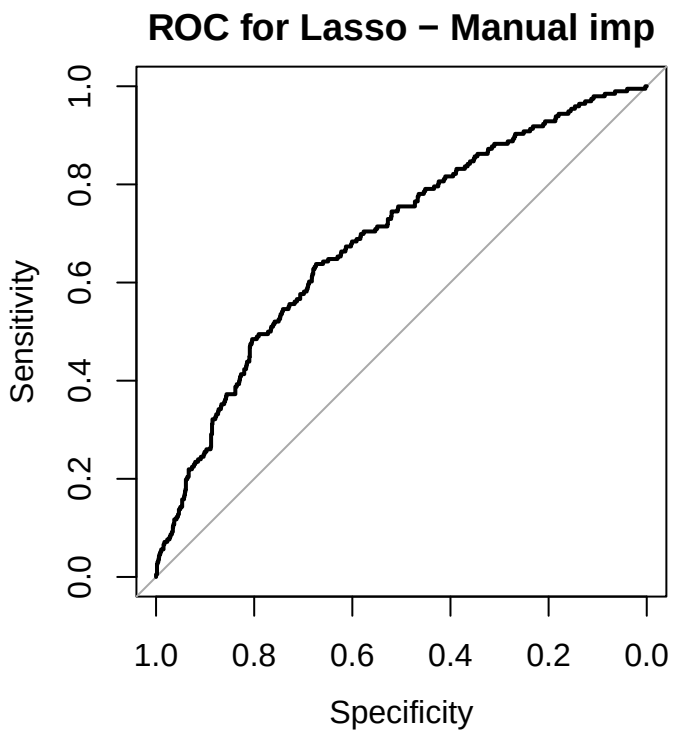
```
plot(cv.out, main = c("Binomial deviance for lambda",""))
```




```
lasso_mod_imp = glmnet(x.train.imp, y.train.imp, family = "binomial", alpha = 1, intercept = F, standar

lasso_coef_imp <- coef(lasso_mod,s=cv.out$lambda.1se)
```

We also give the confusion matrix, the ROC and the ROC-AUC.

```
lasso_preds_imp <- predict(lasso_mod_imp, newx = x.test.imp, type = "response",  s = cv.out$lambda.1se)

predicted_value_imp <- factor(round(lasso_preds_imp))
expected_value_imp <- factor(d.test.imp$TenYearCHD)

conf_mat_imp = confusionMatrix(data=predicted_value_imp, reference = expected_value_imp)$table


roc_obj_imp <- roc(d.test.imp$TenYearCHD, lasso_preds_imp, levels = c(0,1), direction = "<")
plot(roc_obj_imp, main = "ROC for Lasso - Manual imp", cex = 0.5)
```




```
auc(roc_obj_imp)
```

```
## Area under the curve: 0.6881
```

```
conf_mat_imp
```

```
##           Reference
## Prediction   0   1
##          0 593  58
##          1 459 138
```

```
sens_imp = sensitivity(conf_mat_imp)
spes_imp = specificity(conf_mat_imp)
names_lasso_coef_imp = names(which(lasso_coef_imp[,1] > 0))
```

The performance of this model is quite similar to that of the complete case, which is to be expected. The Lasso on the manually imputed data chooses male, age, cigsPerDay, prevalentHyp, sysBP, glucose as the significant covariates.

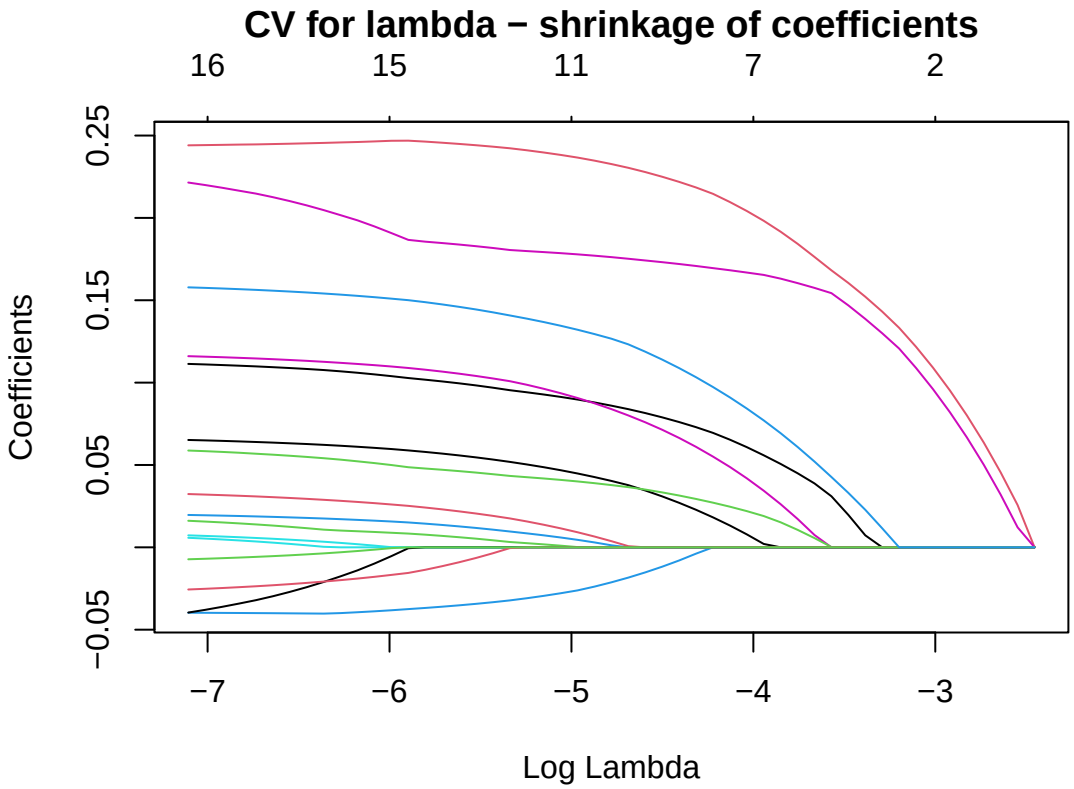
Recall that the sensitivity can be measured by the true positive rate (i.e. the number of true positives over all positives) and the specificity can be measured by the true negative rate (i.e. true negative over all negative). Comparing the Lasso on the complete case data with the imputed data, we note that we have sensitivity 0.575594 in the complete case and sensitivity 0.5636882 in the imputed case. The specificity of the complete case Lasso is 0.7093023, while for the imputed case it is 0.7040816.

Although the difference is not huge, it may resemble that the data quality in the imputed data is slightly lower, although for selecting covariates, we obtained the same answer. In our data rich situation, this

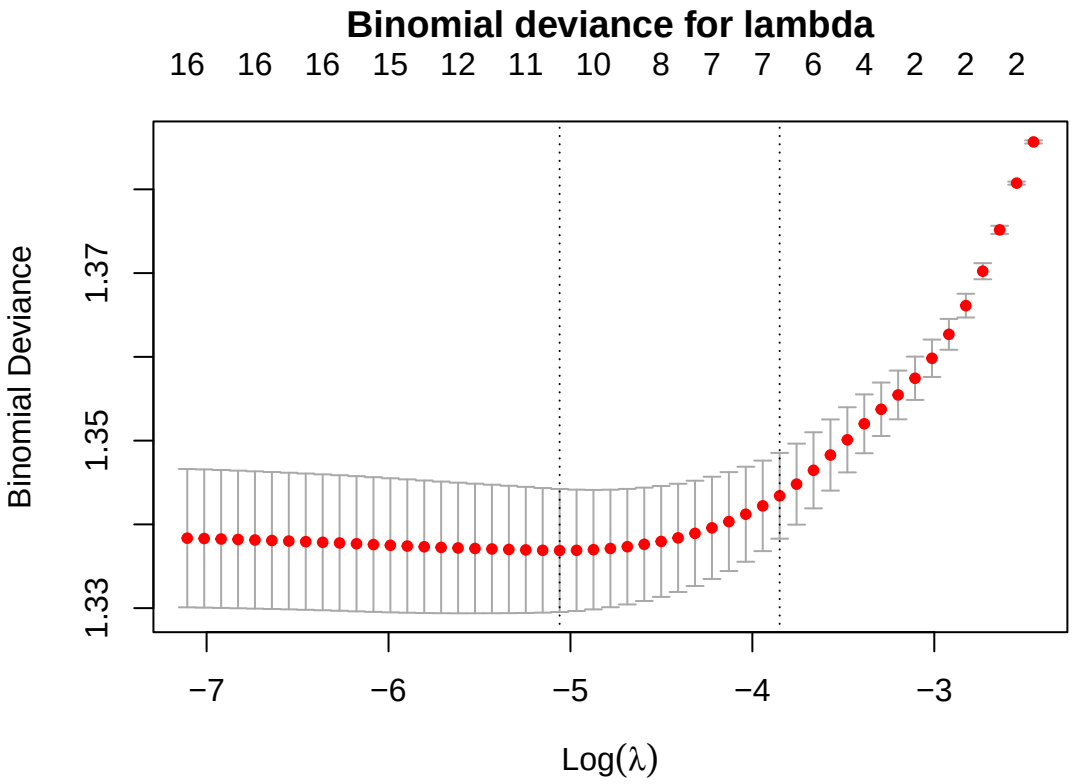
is neither clear enough to be rendered true, nor actually a problem, but for data poor situations, this is something to keep in mind.

We try to do the same thing, using the `MICE`-imputed data set, first plotting the shrinkage and binomial deviance over λ, and then give the confusion matrix, ROC and ROC-AUC.

```
set.seed(8701)
# Cross-validation
cv.out = cv.glmnet(x.train.imp.mice, y.train, family = "binomial", intercept = F, standardize=TRUE, alpl

plot(cv.out$glmnet.fit, "lambda", label=F, main = c("CV for lambda - shrinkage of coefficients",""))
```

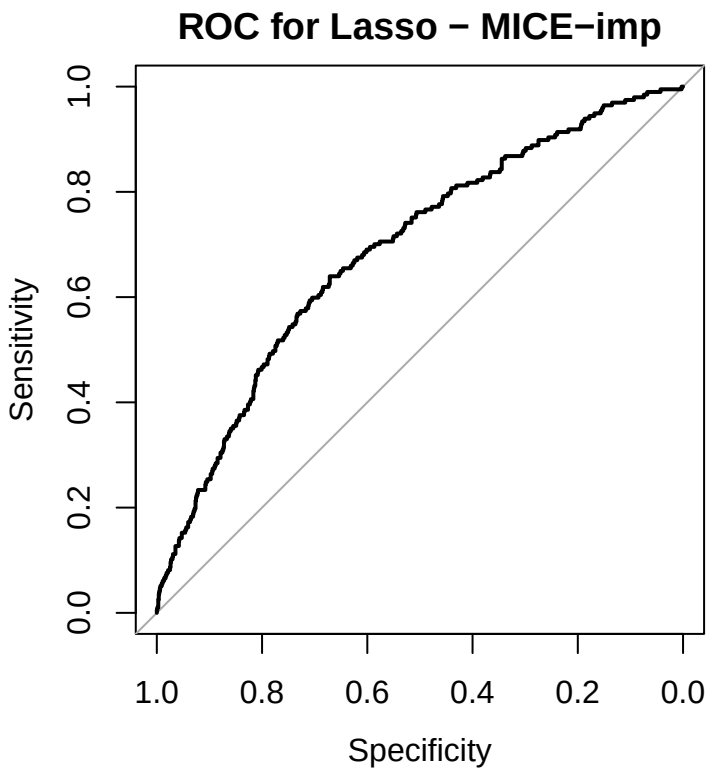



```
plot(cv.out, main = c("Binomial deviance for lambda", ""))
```




```
#cv.out$lambda.min
#cv.out$lambda.1se


lasso_mod_mice = glmnet(x.train.imp.mice, y.train, family = "binomial", alpha = 1, intercept = F, standa

lasso_coef_mice <- coef(lasso_mod_mice,s=cv.out$lambda.1se)
```

```
lasso_preds_mice <- predict(lasso_mod_mice, newx = x.test.imp.mice, type = "response", s = cv.out$lambda

predicted_value_mice <- factor(round(lasso_preds_mice))
expected_value_mice <- factor(y.test)

conf_mat_mice = confusionMatrix(data=predicted_value_mice, reference = expected_value_mice)$table

roc_obj_mice <- roc(y.test, lasso_preds_mice, levels = c(0,1), direction = "<")
plot(roc_obj_mice, main = "ROC for Lasso - MICE-imp", cex = 0.5)
```




```
auc(roc_obj_mice)
```

```
## Area under the curve: 0.6891
```

```
conf_mat_mice
```

```
##           Reference
## Prediction   0   1
##          0 614  58
##          1 460 139
```

```
sens_mice = sensitivity(conf_mat_mice)
spes_mice = specificity(conf_mat_mice)
names_coef_mice = names(which(lasso_coef_mice[,1] > 0))
```

The predictive performance has not changed a lot, in the eyes of AUC. The performance of this model is quite similar to that of the complete case, which is to be expected. The Lasso on the `MICE`-imputed data chooses male, age, cigsPerDay, prevalentHyp, sysBP, glucose as the significant covariates.

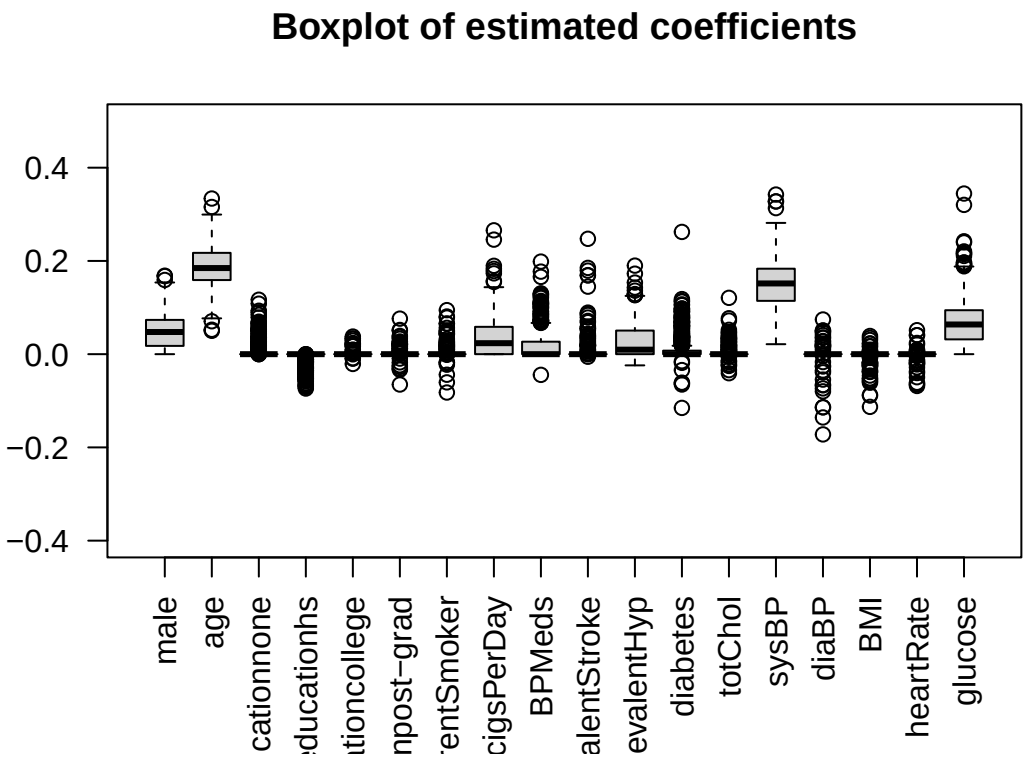
We suspect that the sensitivity and specificity is even lower for the `MICE`-imputed data, as it also imputes those with more than one missing value.

Comparing the Lasso on the complete case data with the imputed data, we note that we have sensitivity 0.5636882 in the manually imputed case and 0.5716946. The specificity of the manually imputed case Lasso is 0.7040816, while for the mice case it is 0.7055838.

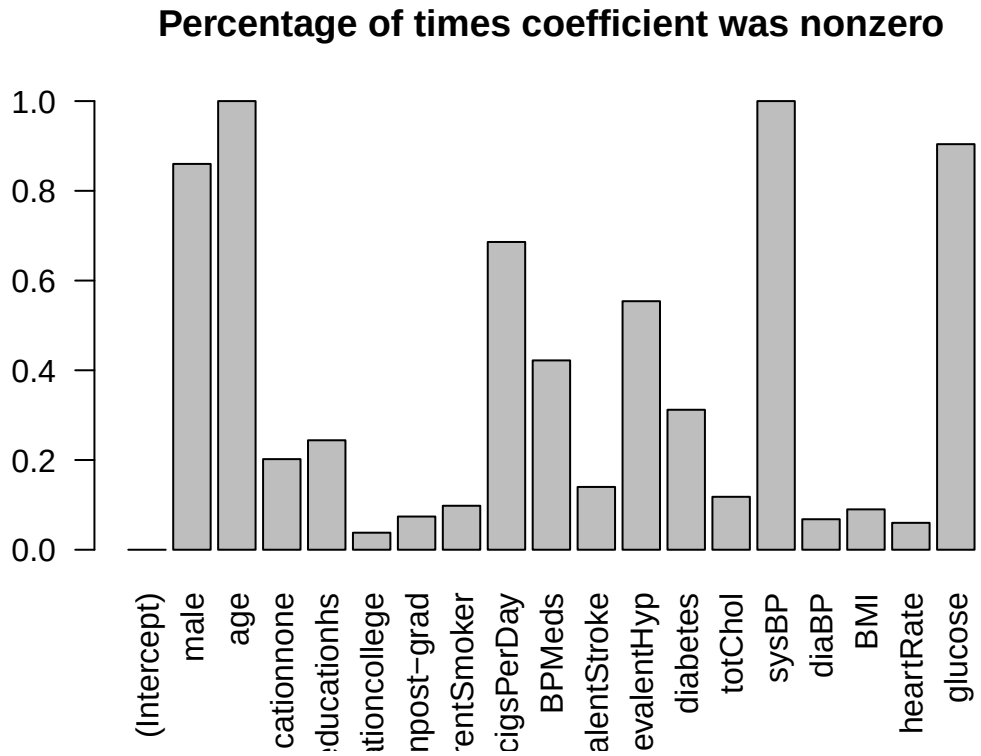
Bootstrapping

To obtain improved estimates for the coefficients in the Lasso, we use bootstrapping. The data set used will be the MICE-imputed data set.

```
set.seed(8701)
B = 500
size = dim(x.train.imp.mice)[1]

B_coef_mice = matrix(NA, nrow = B, ncol = length(lasso_coef_mice[,1]))
for (i in 1:B){
  boot_ind = sample(1:size, size = size, replace = TRUE)

  x = x.train.imp.mice[boot_ind,]
  y = y.train[boot_ind]

  cv.out = cv.glmnet(x, y, family = "binomial", intercept = F, alpha = 1)
  lasso_mod = glmnet(x, y, family = "binomial", intercept = F, alpha = 1, lasso = cv.out$lambda.1se)

  B_coef_mice[i,] <- coef(lasso_mod,s=cv.out$lambda.1se)[,1]
}

colnames(B_coef_mice) = names(coef(lasso_mod,s=cv.out$lambda.1se)[,1])
boxplot.matrix(B_coef_mice[,-1], ylim = c(-0.4,0.5), las = 2, main = "Boxplot of estimated coefficients
```



```
#, ylim = c(-0.02,0.02)

B_coef_count_mice = ifelse(B_coef_mice == 0,0,1)
barplot(apply(B_coef_count_mice, 2, sum)/B, las = 2, main = "Percentage of times coefficient was nonzero
```




```
names_lasso_mice_70 <- names(apply(B_coef_count_mice, 2, sum)/B)[apply(B_coef_count_mice, 2, sum)/B > 0
names_lasso_mice_50 <- names(apply(B_coef_count_mice, 2, sum)/B)[apply(B_coef_count_mice, 2, sum)/B > 0
```

We ran the bootstrap using 500 iterations. The variables that have nonzero coefficients in the Lasso models at least 70% of the time, are male, age, sysBP, glucose. Similarly, by those to those who are included at least 50% of the time, we obtain male, age, cigsPerDay, prevalentHyp, sysBP, glucose. This is indeed similar to the ones we picked out earlier.

Inference

In order to do inference we simply fit a logistic regression model using the `glm` function in `R`, and extract the inference from there. However, we will keep the coefficients chosen by the lasso-bootstrapping iterations in the earlier models, and now use the test data to fit a logistic model to avoid overfitting. Note that we have only used the test data to predict and observe different measures, such as ROC-AUC, sensitivity/specificity, and so on. The test data is therefore suitable for inference, as it has not been perturbed in the procedure of fitting the models.

We start out with the complete case models and fit a logistic model with the most important variables, namely male, age, sysBP, glucose. We first state the coefficients of the new regression model and their confidence intervals.

```r
# Fit the model on the complete case training data.

mod <- glm(TenYearCHD ~ age + male + sysBP + glucose, data = d.complete[-train,], family = binomial())

# Confidence intervals.

CI_mod = confint(mod)

CI_mod0 = confint(mod0)

# Look at the coefficients

summary(mod)$coefficients
```

```
##                Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept) -8.42663685 0.727151909 -11.588551 4.710346e-31
## age          0.04891710 0.011168331   4.379982 1.186889e-05
## male         0.85267388 0.181851299   4.688852 2.747416e-06
## sysBP        0.01998341 0.003974812   5.027511 4.968880e-07
## glucose      0.01273926 0.003233364   3.939941 8.150151e-05
```

The confidence intervals of these variables for the new regression model (left) and naive logistic model fit (right) on the complete data set is given by the following.

```
##                 2.5 %   97.5 %    2.5 %   97.5 %
## (Intercept) -9.89068 -7.03681 -9.90421 -6.51217
## age          0.02715  0.07099  0.04840  0.08067
## male         0.49970  1.21362  0.34667  0.87569
## sysBP        0.01226  0.02786  0.00651  0.02463
## glucose      0.00673  0.01950  0.00216  0.01295
```

We can observe that all the coefficients in the new model are significant! Comparing confidence intervals for the naive model and the new model with Lasso-selected variables, we see that the confidence intervals are shifted more towards zero, and some of them has even become slightly smaller. For example, the confidence interval of `sysBP` went from 0.0181175 to 0.0156049 after the subset selection.

Similarly, we may fit a logistic model on the imputed data. We include the variables (male, age, sysBP, glucose) that was nonzero more than 70% of the times in the bootstrap, and we state the coefficients of the new regression model and their confidence intervals.

```r
# Add the response values that we took out for the mice data.

d.test.imp.mice["TenYearCHD"] <- y.test

# Fit the model.
mod <- glm(TenYearCHD ~ age + male + sysBP + cigsPerDay + glucose, data = d.test.imp.mice, family = bin

# Consider confidence interval and coefficients
CI_mice = confint(mod)
cbind(as.matrix(round(summary(mod)$coefficients,5)), as.matrix(round(CI_mice,5)))
```

```
##             Estimate Std. Error   z value Pr(>|z|)    2.5 %   97.5 %
```

```
## (Intercept) -7.48558    0.68760 -10.88645  0.00000 -8.86105 -6.16226
## age          0.05676    0.01053   5.39008  0.00000  0.03627  0.07759
## male         0.38643    0.17215   2.24479  0.02478  0.04887  0.72451
## sysBP        0.01407    0.00361   3.89278  0.00010  0.00699  0.02119
## cigsPerDay   0.02222    0.00670   3.31703  0.00091  0.00902  0.03534
## glucose      0.00701    0.00280   2.49955  0.01244  0.00146  0.01264
```

Again, we can observe that all the coefficients in the new model are significant!

We take a brief look at the complete case model with Lasso-selected variables to the model on the imputed data with Lasso-selected variables. The intervals are for the model on the imputed data (left) and the model on the complete data (right).

```
cbind(as.matrix(round(CI_mice[c(1,2,3,4,6,5),],4)), rbind(as.matrix(round(CI_mod,4)),c("*","*")))
```

```
##             2.5 %    97.5 %    2.5 %     97.5 %
## (Intercept) "-8.861" "-6.1623" "-9.8907" "-7.0368"
## age         "0.0363" "0.0776"  "0.0272"  "0.071"
## male        "0.0489" "0.7245"  "0.4997"  "1.2136"
## sysBP       "0.007"  "0.0212"  "0.0123"  "0.0279"
## glucose     "0.0015" "0.0126"  "0.0067"  "0.0195"
## cigsPerDay  "0.009"  "0.0353"  "*"       "*"
```

```
diff1 = CI_mod[3,2] - CI_mod[3,1]
diff2 = CI_mice[3,2] - CI_mice[3,1]
```

We can see, for example by considering `sysBP`, see that the width of the interval has gone further down by working on the imputed data, as the confidence interval width for `sysBP` was 0.7139159 for the complete case model and 0.6756384. This may simply be because we use more data to fit the model, but it may also be a nonsensical question, as we are in essence fitting two different models. The inclusion of `cigsPerDay` in the model on imputed data is probably a key reason why we see such a difference. More or less, we obtain the same results, as it is the same covariates that come back time and time again.

Discussion

When we compare the chosen coefficients from the complete case data compared to the imputed data, we see that they correspond. Thus, in a data-rich situation imputation may only introduce unnecessary variance, without having an immediate effect on the quality of the model or inference.

Another interesting thing we discovered, was that even though we obtain good results with the imputed data, the data quality seems to go down, if only barely. The sensitivity and specificity of the model went down slightly when using the imputed data, but as the Lasso selects the same variables, it does not matter a lot for our purposes. The decrease in data quality might have been more visible if the percentage of imputed values were higher, or if we were in a data-poor situation.

What we can probably conclude, is that the variables male, age, sysBP, glucose are the most significant, and that other important variables are `cigsPerDay` and `prevalentHyp`.

References

Bates, Douglas, Martin Maechler, and Mikael Jagan. 2022. *Matrix: Sparse and Dense Matrix Classes and Methods*. https://CRAN.R-project.org/package=Matrix.

Binder, Martin, Florian Pfisterer, Michel Lang, Lennart Schneider, Lars Kotthoff, and Bernd Bischl. 2021. "mlr3pipelines - Flexible Machine Learning Pipelines in r." *Journal of Machine Learning Research* 22 (184): 1–7. https://jmlr.org/papers/v22/21-0281.html.

Binder, Martin, Florian Pfisterer, Lennart Schneider, Bernd Bischl, Michel Lang, and Susanne Dandl. 2022. *Mlr3pipelines: Preprocessing Operators and Pipelines for Mlr3*. https://CRAN.R-project.org/package=mlr3pipelines.

Borowski, Jan, and Piotr Fic. 2022. *NADIA: NA Data Imputation Algorithms*. https://CRAN.R-project.org/package=NADIA.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1–22. https://doi.org/10.18637/jss.v033.i01.

Friedman, Jerome, Trevor Hastie, Rob Tibshirani, Balasubramanian Narasimhan, Kenneth Tay, Noah Simon, and James Yang. 2022. *Glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. https://CRAN.R-project.org/package=glmnet.

Kassambara, Alboukadel. 2022. *Ggcorrplot: Visualization of a Correlation Matrix Using Ggplot2*. http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2.

Kuhn, Max. 2022. *Caret: Classification and Regression Training*. https://github.com/topepo/caret/.

Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. "mlr3: A Modern Object-Oriented Machine Learning Framework in R." *Journal of Open Source Software*, December. https://doi.org/10.21105/joss.01903.

Lang, Michel, Bernd Bischl, Jakob Richter, Patrick Schratz, Martin Binder, Florian Pfisterer, Raphael Sonabend, and Marc Becker. 2022. *Mlr3: Machine Learning in r - Next Generation*. https://CRAN.R-project.org/package=mlr3.

Lang, Michel, Bernd Bischl, Jakob Richter, Xudong Sun, and Martin Binder. 2022. *Paradox: Define and Work with Parameter Spaces for Complex Algorithms*. https://CRAN.R-project.org/package=paradox.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2022. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. https://CRAN.R-project.org/package=e1071.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Robin, Xavier, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. 2011. "pROC: An Open-Source Package for r and s+ to Analyze and Compare ROC Curves." *BMC Bioinformatics* 12: 77.

———. 2021. *pROC: Display and Analyze ROC Curves*. http://expasy.org/tools/pROC/.

Sarkar, Deepayan. 2008. *Lattice: Multivariate Data Visualization with r*. New York: Springer. http://lmdvr.r-forge.r-project.org.

———. 2021. *Lattice: Trellis Graphics for r*. http://lattice.r-forge.r-project.org/.

Simon, Noah, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2011. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software* 39 (5): 1–13. https://doi.org/10.18637/jss.v039.i05.

Tierney, Nicholas, and Dianne Cook. 2023. "Expanding Tidy Data Principles to Facilitate Missing Data Exploration, Visualization and Assessment of Imputations." *Journal of Statistical Software* 105 (7): 1–31. https://doi.org/10.18637/jss.v105.i07.

Tierney, Nicholas, Di Cook, Miles McBain, and Colin Fay. 2023. *Naniar: Data Structures, Summaries, and Visualisations for Missing Data*. https://github.com/njtierney/naniar.

van Buuren, Stef, and Karin Groothuis-Oudshoorn. 2011. "mice: Multivariate Imputation by Chained Equations in r." *Journal of Statistical Software* 45 (3): 1–67. https://doi.org/10.18637/jss.v045.i03.

———. 2022. *Mice: Multivariate Imputation by Chained Equations*. https://CRAN.R-project.org/package=mice.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2022. *Ggplot2: Create Elegant Data Visualisations*

Using the Grammar of Graphics. https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2016. *Bookdown: Authoring Books and Technical Documents with R Markdown*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/bookdown.

———. 2022a. *Bookdown: Authoring Books and Technical Documents with r Markdown*. https://CRAN.R-project.org/package=bookdown.

———. 2022b. *Knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.