Compulsory exercise: Team SuperGreat

MA8701 Advanced Statistical Learning V2023

Nora Aasen, Elias Angelsen, Jonas Nordstrom

28 mars, 2023

Introduction

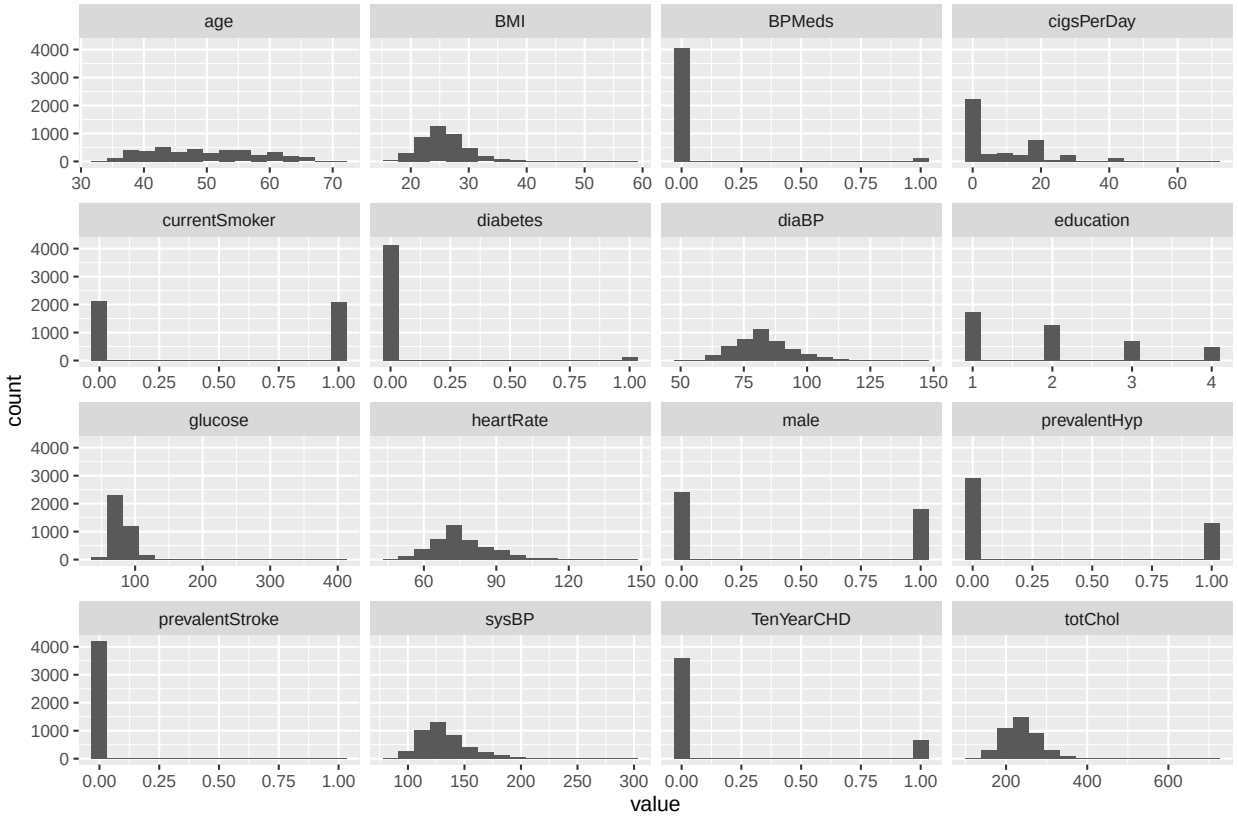
In this project we have studied the Framingham Coronary Heart Disease Dataset which can be found on Kaggle "Logistic Regression to Predict Heart Disease" (n.d.). The dataset contains patient information for inhabitants in Framingham, Massachusetts, and is typically used to predict the chance of getting coronary heart disease (CHD) within the next 10 years.

For this project, however, we intend to investigate how to handle missing data in combination with lasso and bootstrapping as a method for making inference about the variables. We first do single regression imputation using the package called `mice` (van Buuren and Groothuis-Oudshoorn (2022)). The emphasis is on how to avoid data leakage in our procedure. Afterwards we investigate how imputation may affect the subsequent lasso regression, which is done multiple times using bootstrap. Lastly we fit a logistic regression model with the covariates chosen from the lasso procedure(s) using an independent data set, and compare the model based on the imputed and complete data sets.

Exploratory Analysis

We start by examining the data set. We use the packages `ggplot2` (Wickham et al. (2022)) and `tidyr` (Wickham and Girlich (2022)) to plot.

```
# Load and look at data
data <- read.csv("framingham.csv")
data_dim = dim(data)
pos_response = sum(data$TenYearCHD==1)

library(ggplot2)
library(tidyr) # gather()

# 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`, indicating that a natural choice is a logistic regression model. The response variable has 644 positive observations, which equals about 15.2% of the total observations. We see from the plot that most of our covariates are either binary or numeric. However, we note that the variable `education` most likely is 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. We used the packages `ggmice` (Oberman (2022)), `naniar` (Tierney et al. (2023)), and `gridExtra` (Auguie (2017)) to produce the plots.

```
# Look at the missing data
library(ggmice)  # plot_pattern()
library(naniar)  # gg_miss_var()
library(gridExtra)  # grid.arrange()

gluc_miss = sum(is.na(data$glucose))/length(data$glucose)

plot1 <- plot_pattern(data, rotate = T)
plot2 <- gg_miss_var(data)
grid.arrange(plot1, plot2, ncol = 2)
```

As we can see there are seven covariates that contains missing data; `glucose`, `education`, `BPMeds`, `totChol`, `cigsPerDay`, `heartRate`, and `BMI`, where `glucose` is by far the covariate that has the most `NA`'s, with a

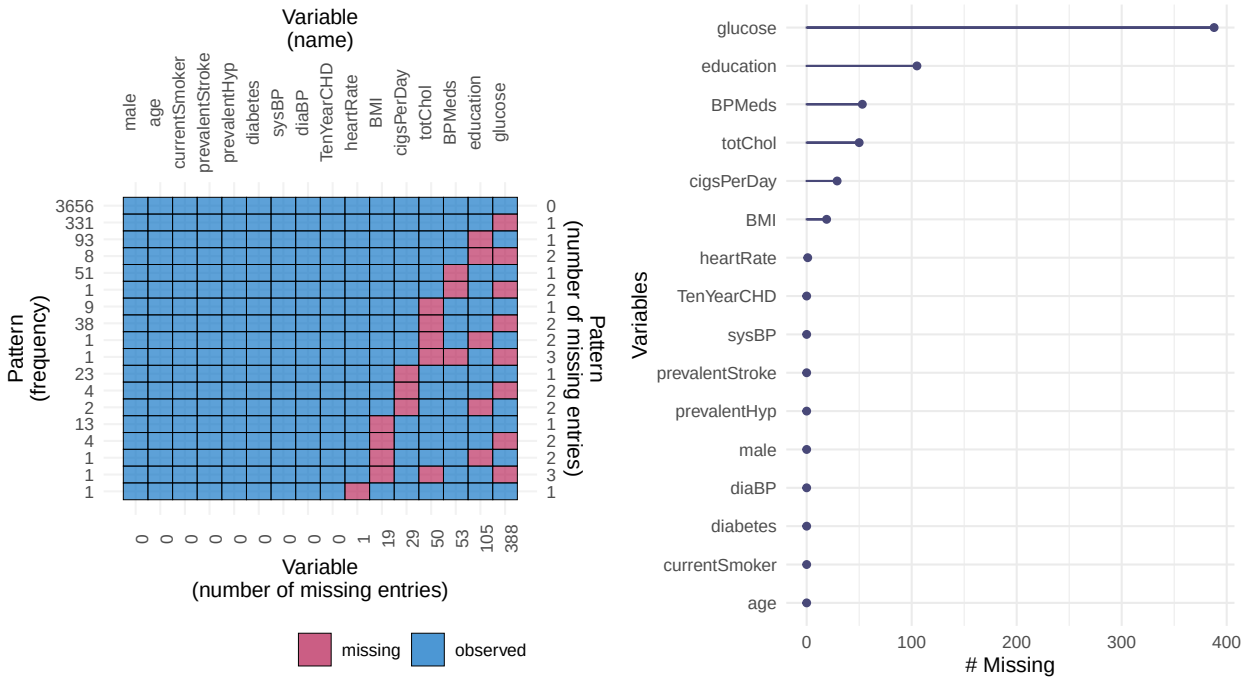


Figure 1: My plot

total of 9.2% missing values. We cannot use the rows that contain missing values as they are. An option for handling the problem would be to remove all rows that contains `NA`'s, also called *complete case analysis*. In this project we use *single imputation*, and compare it with the complete case analysis to investigate how imputing missing data can affect the subsequent inference.

Before beginning the imputation we have to split the data into two disjoint sets. We call them training and test set, but because our goal is inference, they will serve more as two independent sets where one is used for covariate reduction and the other is used for estimation and inference. The same split is used for both the imputed data and the complete data.

```
# Split into training and test first to avoid data leakage
set.seed(8701)
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 positive responses, which equals about 15.2% of the total observations. The proportion of positive observations in the response is the

same, which is a good indicator that our data is missing at random (MAR), which we will come back to.

Missing Data

We categorize missing data into one of three categories: missing completely at random (MCAR), missing at random (MAR), or missing not at random (MNAR). If the data is MCAR, we expect no correlation between the missingness of data and the data itself. Consequently, removing data with missing components should cause no loss in information except the loss due to reduction in data quantity. If the data is MAR, then the structure explaining if the data is missing is dependent on the other variables in our data. In this case, naive methods like mean imputing would cause a loss in information, although we expect methods that predict missing data from the rest of the data to compensate for MAR. For MNAR, the missingness is explained by an unknown source, for instance the unobserved data. In the case of MNAR, any imputation method would yield a biased results if one used only the data at hand (Buuren (2018)).

This can be formulated mathematically as follows. 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. Let R ba an indicator matrix indicating missing (0) and observed (1) covariates. Let ψ be the parameters 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 $P(R|Z, \psi) = P(R|Z, \psi)$.

Hence, we need to investigate the missingess in our data before imputing. One thing we do is compare the frequency of missing data depending on whether the response variable is 0 or 1. We report the number of missing values for each variable and their respective frequencies of missingness for positive/negative responses.

```
# Find number of 0's and 1's and their respective numbers of missing values.
d.pos <- subset(data, TenYearCHD == 1)
d.neg <- subset(data, TenYearCHD == 0)
na.pos <- colSums(apply(d.pos,2,is.na))
na.neg <- colSums(apply(d.neg,2,is.na))

# Construct a table to compare missingness frequency for positive/negative responses.
M = matrix(NA,nrow = 3, ncol = 16)

M[1,] = round(colSums(is.na(data)),0)
M[2,] = round(na.pos/dim(d.pos)[1],3)
M[3,] = round(na.neg/dim(d.neg)[1],3)
colnames(M) = colnames(d.pos)
rownames(M) = c("# miss", "1", "0")
M[,which(colSums(is.na(data)) > 0)]
```

```
##        education cigsPerDay BPMeds totChol    BMI heartRate glucose
## # miss   105.000     29.000 53.000  50.000 19.000     1.000 388.000
## 1          0.025      0.003  0.017   0.014  0.016     0.002   0.078
## 0          0.025      0.008  0.012   0.011  0.003     0.000   0.094
```

Some of our missing data seem dependent on the response, such as `BMI`, `cigsPerDay`, and `glucose`, and this indicates that we should include the response variable in our imputation model. If the goal was prediction,

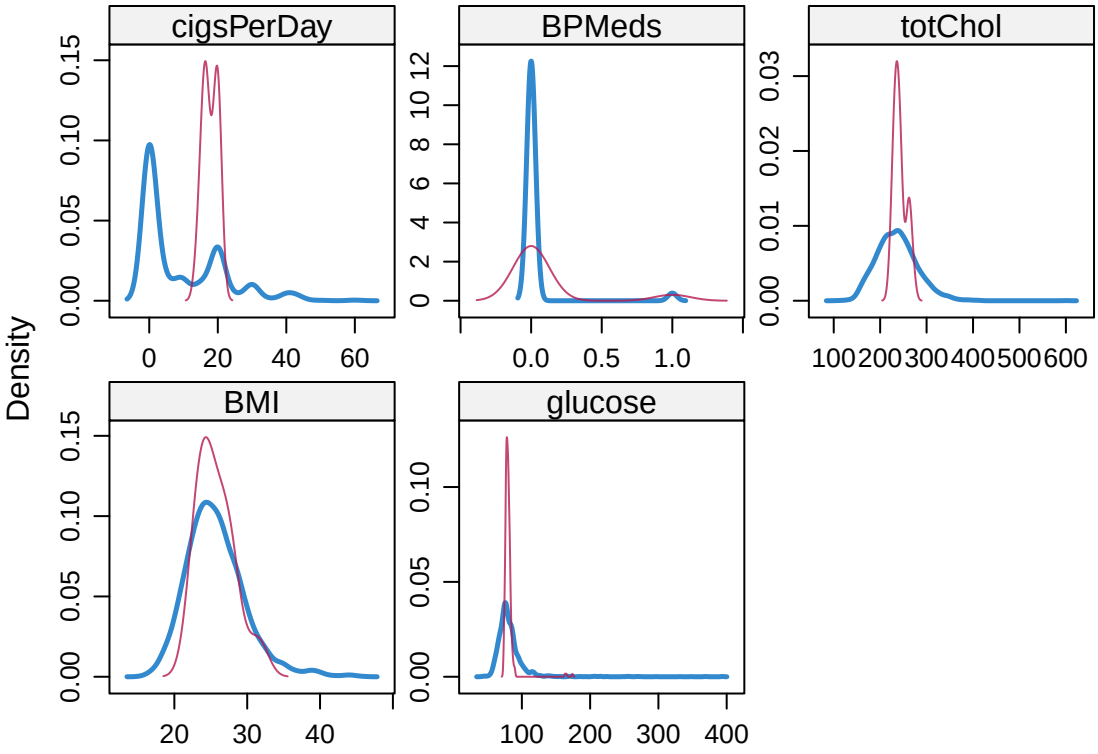
this is not ideal as we use the response to complete the dataset. However, as our aim is inference, we include the response.

We now attempt to predict the missing values using the observed data. A better method for handling missing data is multiple imputation. However, it is not trivial to combine Rubin's rules with the Lasso, bootstrap and the concluding inference. It was therefore not considered for this project.

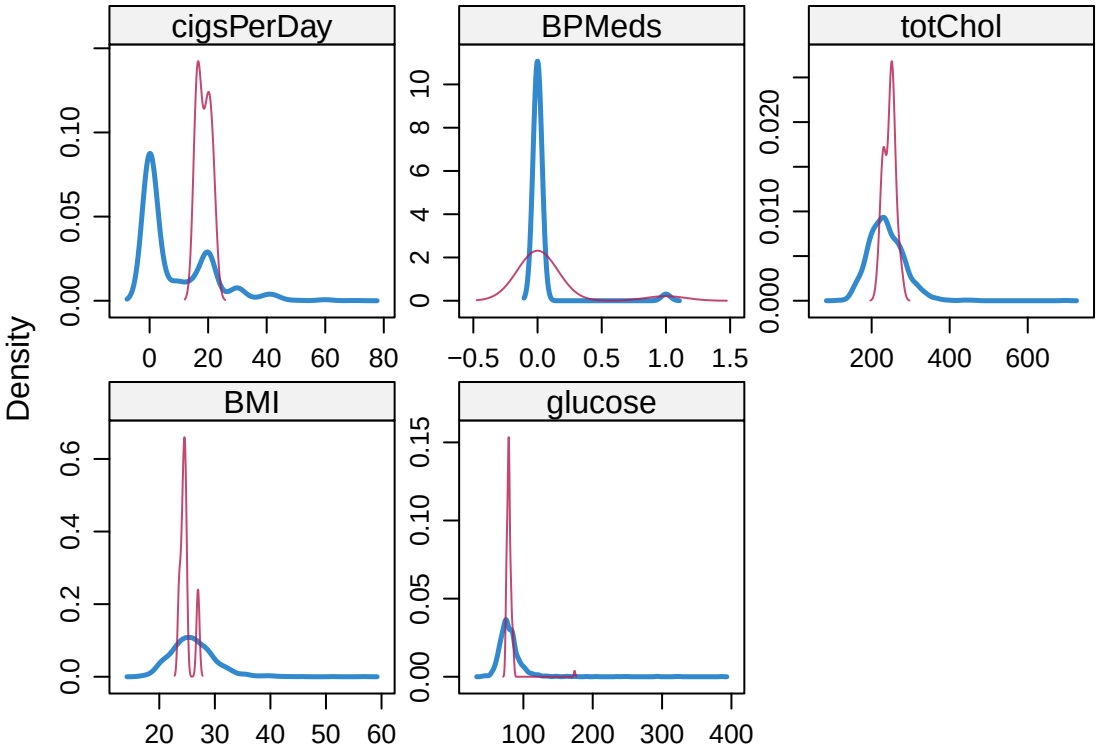
When doing single imputation, the package `mice` (van Buuren and Groothuis-Oudshoorn (2022)) has as default polynomial regression for the factor variable and predictive mean matching (pmm) for all other. We change to logistic regression for the binary variable `BPMeds`, keep polynomial regression for `education` and use linear regression for all other missing variables.

If we wanted to use the test set as normal, we would have to fit the imputation models on the training set, and then reuse these models to impute the test set in order to avoid data leakage. Correlation between the samples in the test set when used for validation or prediction, is unwanted. This can easily be done using the function `mice.reuse` from the package `NADIA` (Borowski and Fic (2022)). However, since we will use both sets for model fitting, altough we (misleadingly) called them train and test, they should be imputed separately with individual models. This is to avoid correlation between the data sets. We also report plots showing the density of imputed values versus the original data.

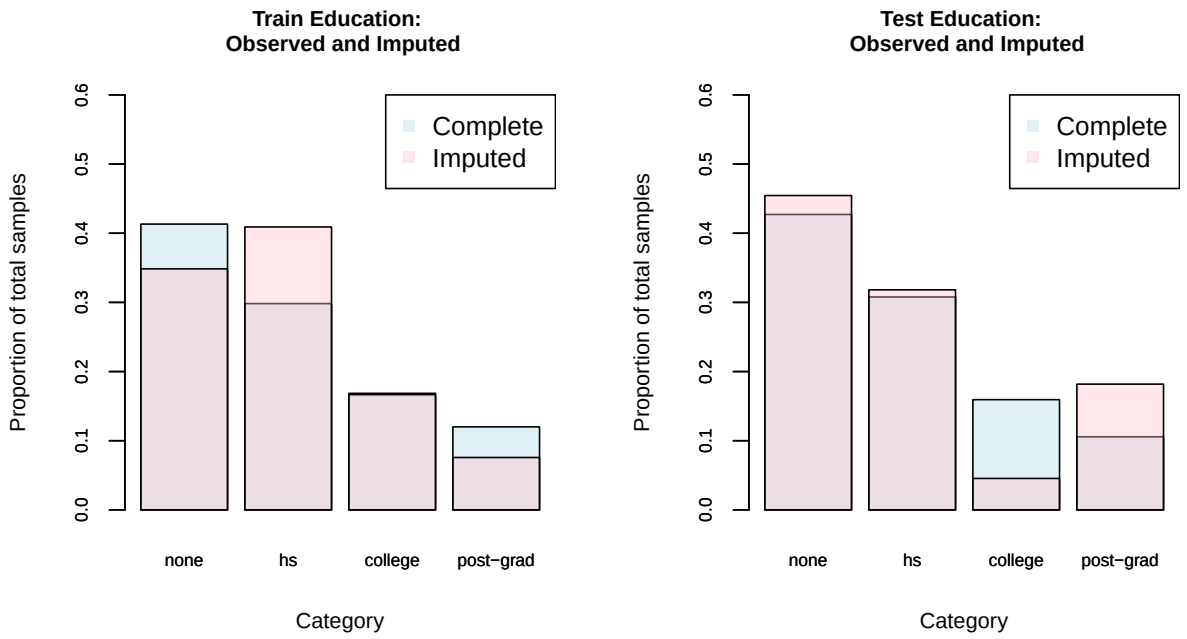
```
library(mice)  # mice()

# Change the method for each covariate to our choice
meth = mice(data, maxit = 0)$method
meth[which(meth == "pmm")] = "norm.predict"
meth["BPMeds"] <- "logreg"

# Make a single imputation model using the training data
imp_mod.train = mice(d.train, m = 1, printFlag = F, method = meth)
d.train.imp = complete(imp_mod.train)
imp_mod.test = mice(d.test, m = 1, printFlag = F, method = meth)
d.test.imp = complete(imp_mod.test)

# Compare the density of imputed data vs. actual data in the training set
densityplot(imp_mod.train)
```




```
densityplot(imp_mod.test)
```




```
par(mfrow = c(1,2),cex.main = 0.8, cex.lab = 0.8, cex.axis = 0.7)
c1 <- rgb(173,216,230, max = 255, alpha = 95, names = "lt.blue")
c2 <- rgb(255,192,203, max = 255, alpha = 95, names = "lt.pink")
barplot(prop.table(table(d.train.complete$education)), col = c1, xlab = "Category",
        ylab = "Proportion of total samples",
        main = "Train Education: \n Observed and Imputed",
        ylim = c(0,0.6)) # Plot 1st histogram using a transparent color
barplot(prop.table(table(d.train.imp$education[is.na(d.train$education)])),
        col = c2, add = TRUE)
legend("topright", legend=c("Complete", "Imputed"), cex=1, pch=15,col=c(c1,c2))

barplot(prop.table(table(d.test.complete$education)), col = c1, xlab = "Category",
        ylab = "Proportion of total samples",
        main = "Test Education: \n Observed and Imputed",
        ylim = c(0,0.6)) # Plot 1st histogram using a transparent color
barplot(prop.table(table(d.test.imp$education[is.na(d.train$education)])),
        col = c2, add = TRUE)
legend("topright", legend=c("Complete", "Imputed"), cex = 1, pch=15,col=c(c1,c2))
```



As can be seen from the density plots, our imputation is mostly yielding imputations around the mean, but with some added variance. Both sets have some deviance from the observed proportions when it comes to imputation of the categorical variable `education`, which we are unsure how to explain. Furthermore, although the imputation of `cigsPerDay` may appear strange, it makes sense as all missing observations in that variable also had responded positive to `currentSmoker`. The imputation of `cigsPerDay` seems to be about the mean of non-zero values, thus showcasing a similar behavior as the other imputed data.

Model

In this section we fit the same model on both the imputed and complete training data. The two data sets are identical apart from the rows that contained `NA`'s. Since we want to do lasso, we first standardize the data. Secondly, we use bootstrap to calculate the non-zero coefficients multiple times. Lastly, we fit a logistic regression model using the test set.

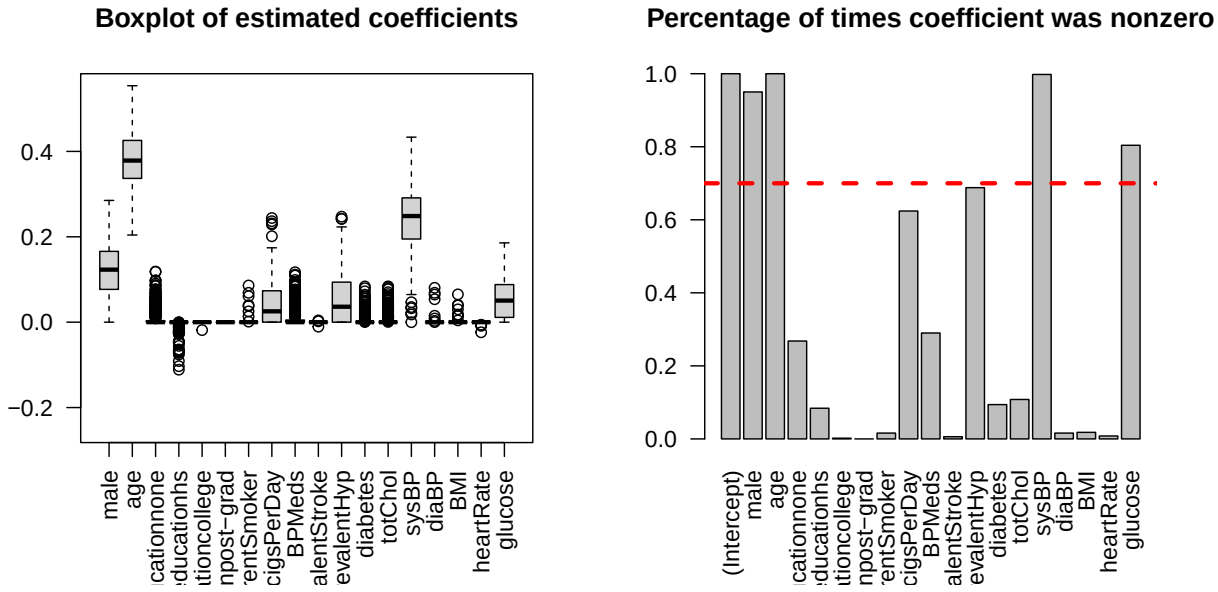
```
# Scale data for lasso
x.train.complete = scale(model.matrix(TenYearCHD ~ . -1,
                                      data = d.train.complete,
                                      family = binomial()))
train.complete.mean = attr(x.train.complete, "scaled:center")
train.complete.sd = attr(x.train.complete, "scaled:scale")
y.train.complete = d.train.complete$TenYearCHD

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
```

Lasso on complete data

We continue to do the Lasso on the complete case data. This is done by first estimating λ_{min} using cross-validation and then fitting a Lasso model with the highest λ within one standard deviation of λ_{min} as penalty term. For easy implementation of the lasso method, we use the package `glmnet` (Friedman et al. (2022)). This is combined with bootstrapping to see which covariates are most often included in the selection process.

```
library(glmnet) # implementing lasso

B = 500 # Number of boostrap datasets
B.size.complete = dim(d.train.complete)[1] # Number of samples
# Make a matrix that store the computed coefficients
B.coef.complete = matrix(NA, nrow = B, ncol = length(colnames(x.train.complete))+1)

for (i in 1:B){
  B.data = sample(1:B.size.complete, size = B.size.complete, replace = TRUE)
  x = x.train.complete[B.data,]
  y = y.train.complete[B.data]
  cv.out = cv.glmnet(x, y, family = "binomial", alpha = 1)
  lasso_mod = glmnet(x, y, family = "binomial", alpha = 1,
                     intercept = T, lasso = cv.out$lambda.1se)

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

```
colnames(B.coef.complete) = names(coef(lasso_mod,s=cv.out$lambda.1se)[,1])
B.coef.count.complete = ifelse(B.coef.complete == 0,0,1)
coef70.complete <- names(apply(B.coef.count.complete,
                               2, sum)/B)[apply(B.coef.count.complete, 2, sum)/B > 0.7]

# We do not include intercept as it is always in the model
# and too large to fit in the scaling.
par(mfrow = c(1,2))
boxplot.matrix(B.coef.complete[,-1], ylim = c(-0.25,0.55),
               las = 2,
               main = "Boxplot of estimated coefficients")
barplot(apply(B.coef.count.complete, 2, sum)/B,
        las = 2,
        main = "Percentage of times coefficient was nonzero")
abline(h=0.7,col="red", lty=2, lwd=3)
```




```
# Fit a logistic model using the test data and chosen covariates
mod.complete <- glm(TenYearCHD ~ age + male + sysBP + glucose,
                    data = d.test.complete, family = binomial())
```

After using Lasso on 500 data sets, the variables that have nonzero coefficients at least 70% of the time, are (Intercept), male, age, sysBP, glucose. They are also indicated on the right figure above. This cut-off value seems reasonable, but could be altered to include more or fewer coefficients if wanted, as it was chosen a bit arbitrarily. We use the chosen covariates to fit a logistic regression model. To avoid overfitting, we use the independent test data to estimate the coefficients in the logistic model.

We could also investigated the box-plot to determine the covariates that we would like in our model. Both `cigsPerDay` and `prevelentHyp` has coefficients estimated to be non-zero, although zero is included in their inter-quartile ranges, which perhaps justifies excluding them from the final model. However, zero is also included in the inter-quartile range of the coefficient of `glucose`, which is included in the final model. Since the choice of a 70% cutoff range is somewhat arbitrary, one could perhaps use both plots in combination to determine the most significant covariates.

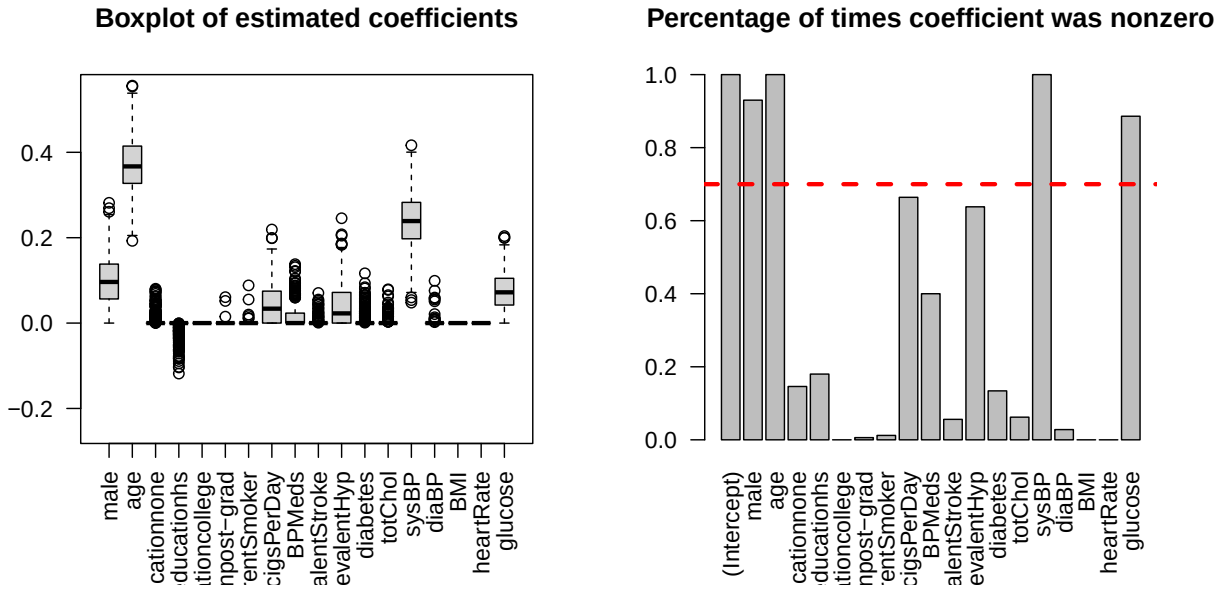
Lasso on imputed data

We now do the same thing, just using the imputed data instead of the complete case.

```
B = 500 # Number of boostrap datasets
B.size.imp = dim(d.train.imp)[1] # Number of samples
# Make a matrix that store the computed coefficients
B.coef.imp = matrix(NA, nrow = B, ncol = length(colnames(x.train.imp))+1)

for (i in 1:B){
  B.data = sample(1:B.size.imp, size = B.size.imp, replace = TRUE)
  x = x.train.imp[B.data,]
  y = y.train.imp[B.data]
  cv.out = cv.glmnet(x, y, family = "binomial", alpha = 1)
  lasso_mod = glmnet(x, y, family = "binomial", alpha = 1,
```

```
                    intercept = T, lasso = cv.out$lambda.1se)

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

```
colnames(B.coef.imp) = names(coef(lasso_mod,s=cv.out$lambda.1se)[,1])
B.coef.count.imp = ifelse(B.coef.imp == 0,0,1)
coef70.imp <- names(apply(B.coef.count.imp,
                          2, sum)/B)[apply(B.coef.count.imp, 2, sum)/B > 0.7]

# We do not include intercept as it is always in the model
# and too large to fit in the scaling.
par(mfrow = c(1,2))
boxplot.matrix(B.coef.imp[,-1], ylim = c(-0.25,0.55), las = 2,
               main = "Boxplot of estimated coefficients")
barplot(apply(B.coef.count.imp, 2, sum)/B, las = 2,
        main = "Percentage of times coefficient was nonzero")
abline(h=0.7,col="red", lty=2, lwd=3)
```




```
# Fit a logistic model using the test data and chosen covariates
mod.imp <- glm(TenYearCHD ~ age + male + sysBP + glucose + cigsPerDay,
               data = d.test.imp, family = binomial())
```

The Lasso on the imputed data chooses (Intercept), male, age, sysBP, glucoseas the non-zero covariates, indicated in the right plot above. The performance of this model is quite similar to that of the complete case, but we note in particular that `cigsPerDay`, `BPMeds` and `glucose` are non-zero more often when using the imputed data. This could perhaps be a sign of correlation in the training data caused by the imputation. As before, we use the chosen covariates to fit a logistic regression model. To avoid overfitting, we use the independent (imputed) test data to estimate the coefficients. This explains why it was important to impute the test set using itself, and not the imputation model trained on the training data. However, in a regular train/test situation where the test set is indeed used for testing or validation, it is important to reuse the imputation model fitted on the training data, as discussed earlier.

Inference

To access the two models we compute the AIC and look at the confidence intervals from the model.

```
data.frame("Complete"=round(summary(mod.complete)$aic,2),
           "Imputed" = round(summary(mod.imp)$aic,2))
```

```
##   Complete Imputed
## 1   887.96 1020.52
```

```
summary(mod.complete)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ age + male + sysBP + glucose, family = binomial(),
##     data = d.test.complete)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5252  -0.6070  -0.4669  -0.3512   2.6205
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.365555   0.714477 -10.309  < 2e-16 ***
## age          0.053072   0.010934   4.854 1.21e-06 ***
## male         0.524847   0.175063   2.998  0.00272 **
## sysBP        0.015454   0.003860   4.003 6.25e-05 ***
## glucose      0.007395   0.002844   2.600  0.00932 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 953.22  on 1097  degrees of freedom
## Residual deviance: 877.96  on 1093  degrees of freedom
## AIC: 887.96
##
## Number of Fisher Scoring iterations: 5
```

```
summary(mod.imp)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ age + male + sysBP + glucose + cigsPerDay,
##     family = binomial(), data = d.test.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5042  -0.6073  -0.4703  -0.3366   2.6708
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -7.541317   0.690304 -10.925  < 2e-16 ***
## age          0.056881   0.010542   5.396 6.82e-08 ***
## male         0.383753   0.172135   2.229 0.025789 *
## sysBP        0.014079   0.003616   3.893 9.89e-05 ***
## glucose      0.007581   0.002820   2.689 0.007173 **
## cigsPerDay   0.022524   0.006722   3.351 0.000806 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1096.3  on 1270  degrees of freedom
## Residual deviance: 1008.5  on 1265  degrees of freedom
## AIC: 1020.5
##
## Number of Fisher Scoring iterations: 5
```

```
confint(mod.complete)
```

```
##                    2.5 %      97.5 %
## (Intercept) -8.797630570 -5.99288133
## age          0.031765493  0.07467830
## male         0.183045486  0.87024058
## sysBP        0.007921332  0.02308639
## glucose      0.001797197  0.01315057
```

```
confint(mod.imp)
```

```
##                    2.5 %      97.5 %
## (Intercept) -8.922624298 -6.21325147
## age          0.036367391  0.07773629
## male         0.046200090  0.72179712
## sysBP        0.006994786  0.02120085
## glucose      0.002045020  0.01327780
## cigsPerDay   0.009282114  0.03568440
```

To account for the cherry-picking of variables that the Lasso causes, we may calculate stricter p-values by multiplying them by the number of covariates chosen for the final model (Ruben Dezeure and Meinshausen (2015)).

```
summary(mod.complete)$coefficients[, 4] * length(summary(mod.complete)$coefficients[,
    4])
```

```
##  (Intercept)          age         male        sysBP      glucose
## 3.207834e-24 6.050596e-06 1.358597e-02 3.123052e-04 4.660663e-02
```

```
summary(mod.imp)$coefficients[, 4] * length(summary(mod.imp)$coefficients[, 4])
```

```
##  (Intercept)          age         male        sysBP      glucose   cigsPerDay
## 5.273971e-27 4.094036e-07 1.547359e-01 5.935507e-04 4.303852e-02 4.835542e-03
```

All coefficients continue to be significant after the adjustment for the complete model, although `glucose` is only barely so. For the imputed dataset, only `male` fails to be significant after the adjustment. This is a somewhat surprising result, since `male` where almost always included by the Lasso. Perhaps the absence of `cigsPerDay` and the high correlation between this covariate and `male`, which is 0.331, weighted `male` as more significant covariate than it should have been.

We see that the AIC evaluates the model fitted on imputed data to be worse. This could be penalty for adding an extra coefficient in our model, so out of curiosity we compare with a model fitted using the imputed test set, but with the same covariates as chosen by the Lasso on the complete data.

```
# Check the AIC when using the same covariates
mod.imp2 = glm(TenYearCHD ~ age + male + sysBP + glucose,
               data = d.test.imp, family = binomial())
summary(mod.imp2)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ age + male + sysBP + glucose, family = binomial(),
##     data = d.test.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4771  -0.6098  -0.4754  -0.3570   2.5892
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.017429   0.661384 -10.610  < 2e-16 ***
## age          0.050017   0.010175   4.916 8.85e-07 ***
## male         0.553023   0.162609   3.401 0.000672 ***
## sysBP        0.014043   0.003599   3.902 9.56e-05 ***
## glucose      0.007182   0.002780   2.583 0.009788 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1096.3  on 1270  degrees of freedom
## Residual deviance: 1019.5  on 1266  degrees of freedom
## AIC: 1029.5
##
## Number of Fisher Scoring iterations: 5
```

```
confint(mod.imp2)
```

```
##                    2.5 %      97.5 %
## (Intercept) -8.339200401 -5.74325610
## age          0.030173778  0.07010211
## male         0.235243686  0.87343100
## sysBP        0.006995738  0.02113427
## glucose      0.001680692  0.01277382
```

```r
# Compare the AIC for each model
data.frame("Complete"=round(summary(mod.complete)$aic,2),
      "Imputed" = round(summary(mod.imp)$aic,2),
      "Alternative imputed" = round(summary(mod.imp2)$aic,2))
```

```
##   Complete Imputed Alternative.imputed
## 1   887.96 1020.52             1029.51
```

```r
# Computing the width of the confidence intervals for each variable in each model.
CI.mat = matrix(NA, nrow = 3, ncol = 6)
CI.mat[1,] = c(round(confint(mod.complete)[,1]-confint(mod.complete)[,2],2),NA)
CI.mat[2,] = c(round(confint(mod.imp)[,1]-confint(mod.imp)[,2],2))
CI.mat[3,] = c(round(confint(mod.imp2)[,1]-confint(mod.imp2)[,2],2),NA)
colnames(CI.mat) = names(coef(mod.imp))
rownames(CI.mat) = c("Complete", "Imputed", "Alternative Imputed")
abs(CI.mat)
```

```
##                     (Intercept)  age male sysBP glucose cigsPerDay
## Complete                   2.80 0.04 0.69  0.02    0.01         NA
## Imputed                    2.71 0.04 0.68  0.01    0.01       0.03
## Alternative Imputed        2.60 0.04 0.64  0.01    0.01         NA
```

There is almost no change in AIC between the two models used on the imputed data set, indicating that it is the imputation procedure that causes the AIC to evaluate this model as worse, despite increasing the size of the data. However, from the table above, which shows the width of the 95% confidence interval for the estimated coefficients, we note that the confidence intervals are slightly smaller for the imputed models. This is likely only due to the fact that the imputed data set is larger, and that few of our imputed values deviated much from the mean. Therefore, the smaller confidence intervals do not necessarily imply that these are more precise estimates of the true coefficients.

Discussion

When we compare the coefficients chosen by the Lasso on the complete case data and the imputed data, we see that they correspond. Thus, in a data-rich situation, imputation may end up only contributing to an unnecessary change in the variance, without having an immediate effect on the quality of the model or the inference. This was exemplified in the bootstrap estimates of the lasso regression on imputed data, where covariates where deemed more significant after imputing than before.

Furthermore, the problem with data leakage, both between independent sets used for fitting models, training and test sets used for fitting and validation, and within a single observation for instance between response and covariates, makes imputation vulnerable for unintended inaccuracies. Note that we did not use ROC-AUC to evaluate the models, as both the training and test data had been used to fit the final model. Hence, a third, independent subset would have been needed in order to run ROC-AUC calculations.

References

Auguie, Baptiste. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. https://CRAN.R-project.org/package=gridExtra.

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

Buuren, Stef van. 2018. "Flexible Imputation of Missing Data." 2018. https://stefvanbuuren.name/fimd/.
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.
"Logistic Regression to Predict Heart Disease." n.d. Accessed March 26, 2023. https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression.
Oberman, Hanne. 2022. *Ggmice: Visualizations for Mice with Ggplot2*. https://CRAN.R-project.org/package=ggmice.
Ruben Dezeure, Lukas Meier, Peter Bühlmann, and Nicolai Meinshausen. 2015. "High-Dimensional Inference: Confidence Intervals, p-Values and r-Software Hdi." https://doi.org/10.1214/15-STS527.
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. 2022. *Mice: Multivariate Imputation by Chained Equations*. https://CRAN.R-project.org/package=mice.
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, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.