Compulsory exercise: Team SuperGreat

MA8701 Advanced Statistical Learning V2023

Nora Aasen, Elias Angelsen, Jonas Nordstrom

27 mars, 2023

Introduction

Possibly elaborate some, both concerning presentation of dataset and summary of our project

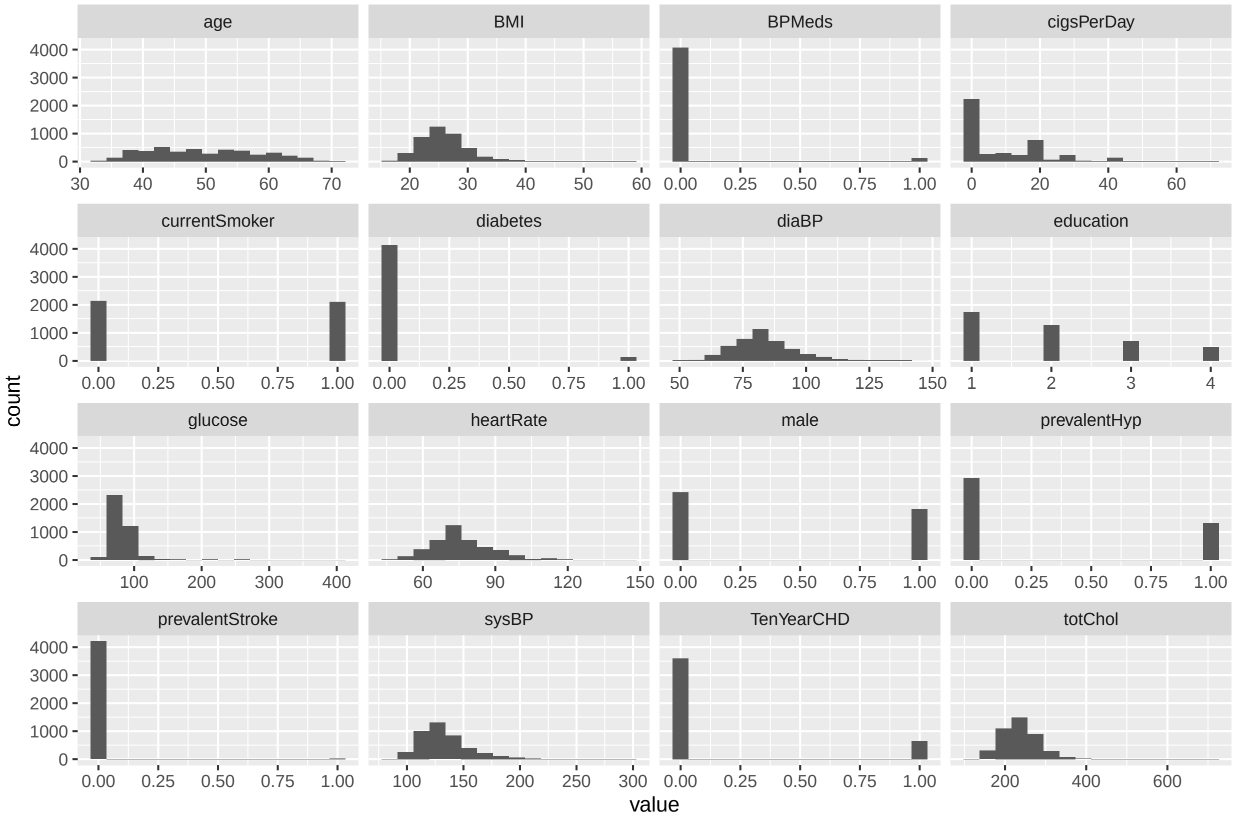
In this project we have studied the Framingham Coronary Heart Disease Dataset which can be found on kaggle "Logistic Regression to Predict Heart Disease" (n.d.). This dataset contains patient information for inhabitants in Framingham, Massachusetts, and is typically used to predict the chance of getting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

For this project, however, we intend to investigate how to handle missing data in combination with lasso as a method for making inference about the variables. We will do single regression imputation using the package called `mice` van Buuren and Groothuis-Oudshoorn (2022), and investigate how imputation may affect the subsequent lasso regression.

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 make the plots.

```
# 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)

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 model 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 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. 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 total

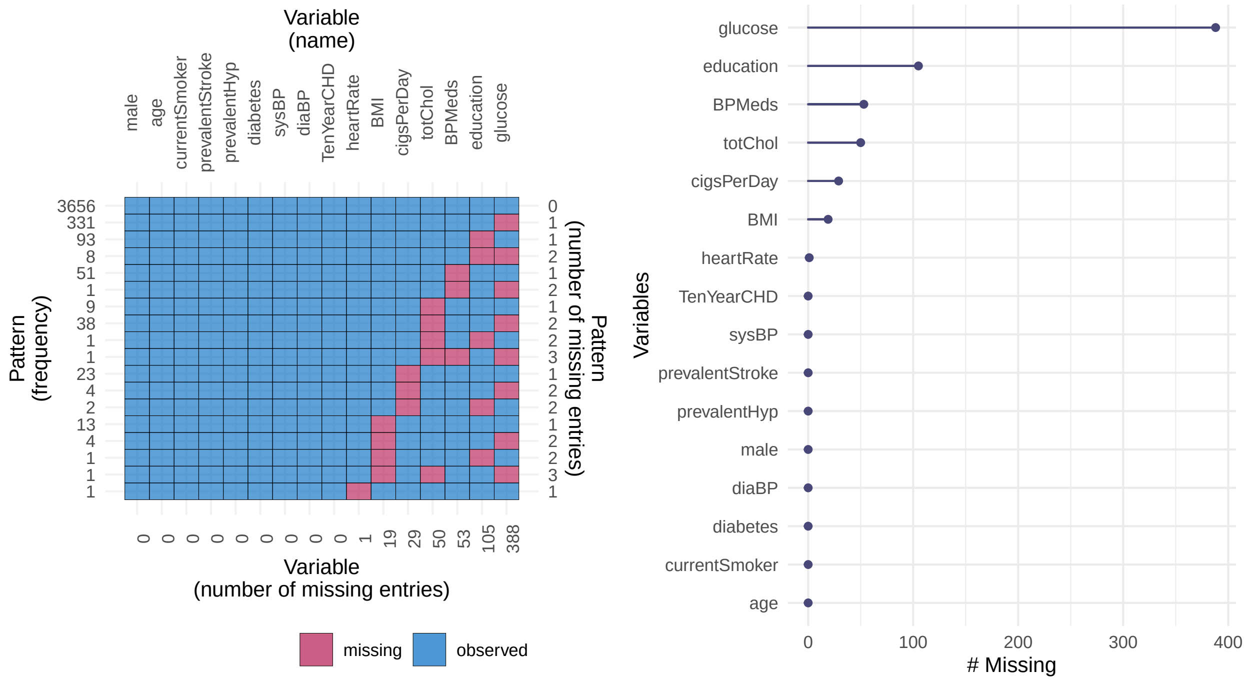


Figure 1: My plot

of 9.2% missing values. We cannot use the rows that contain missing values as is. 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 do a method called *single imputation*, and compare it with the complete case analysis to investigate how imputing missing data can affect the subsequent inference.

See Figure @ref(fig:plot_missing_data).

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

```
# 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. 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), which we will come back to.

Missing Data: Elias kan få skrive denne delen!

Mangler å renskrive teorien, kommentere på MAR; MCAR; or MNAR, kommentere fordelingen etter imputeringen. Evt. se mail med kommentarer for inspo til hva som bør med. Prøv å begrense mengden tekst så langt det lar seg gjøre.

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).

```r
# We investigate the response variable for rows that has missing data
x = which(colSums(is.na(data)) > 0)
M = matrix(nrow=2, ncol = length(x)+1)

# For row 1 we see how many rows that has NA for each covariate
M[1,] = c(colSums(is.na(data))[x], NA)

# For row 2 we compute the percentage of positive responses only for those rows that has NA in that cov
for (i in 1:length(x)){
  r = is.na(data[,x[i]])
  df = data[r,]
  M[2,i] = round(sum(df$TenYearCHD)/length(df$TenYearCHD),3)
}

M[2,length(x)+1] = round(sum(data$TenYearCHD)/length(data$TenYearCHD),3)
colnames(M) = c(colnames(data)[colSums(is.na(data)) > 0],"full")
rownames(M) = c("# miss", "response freq")
as.table(M)
```

```
##                education cigsPerDay  BPMeds totChol     BMI heartRate glucose
## # miss          105.000     29.000  53.000  50.000  19.000     1.000 388.000
## response freq     0.152      0.069   0.208   0.180   0.526     1.000   0.129
##                    full
## # miss
## response freq   0.152
```

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.

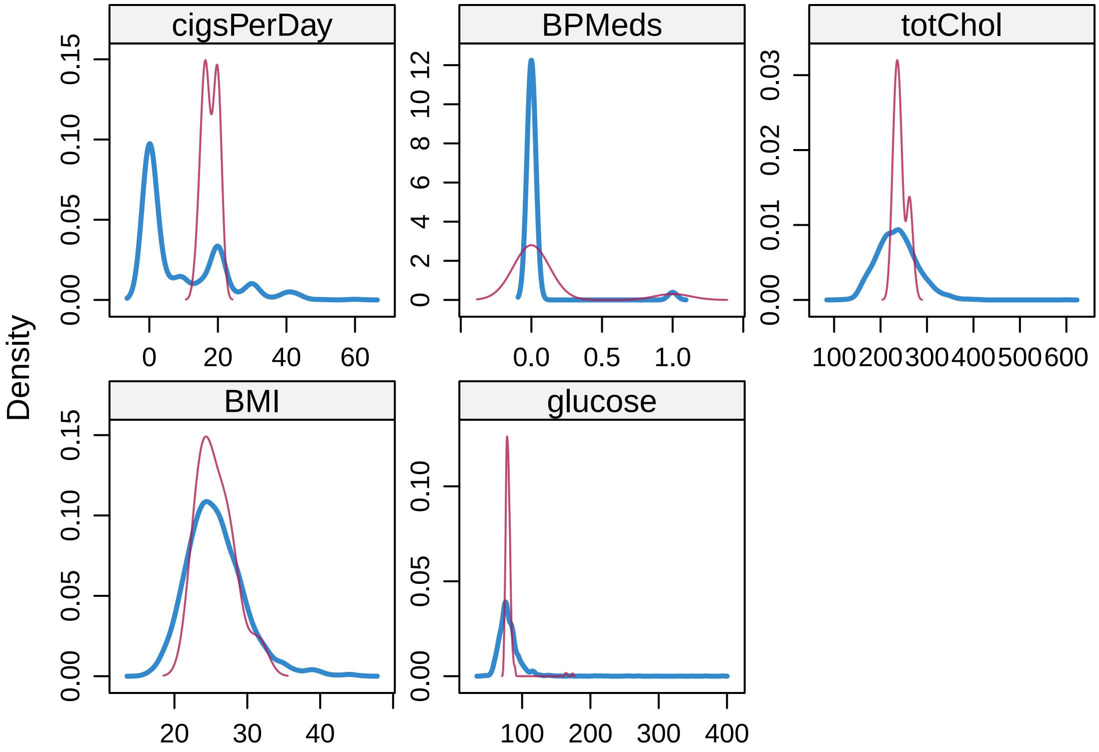
Our data is split into training and test sets, with the test-to-training ratio being 3:7. In order to avoid data leakage in our imputation of the test set, we fit the imputation models on the training set. The main idea is that the test set should be viewed as several independent observations. Using the test set to impute itself will use information not present at the time of training and will yield unintended correlation.

Mice uses polyreg for the factor variable and predictive mean matching (pmm) for all other.

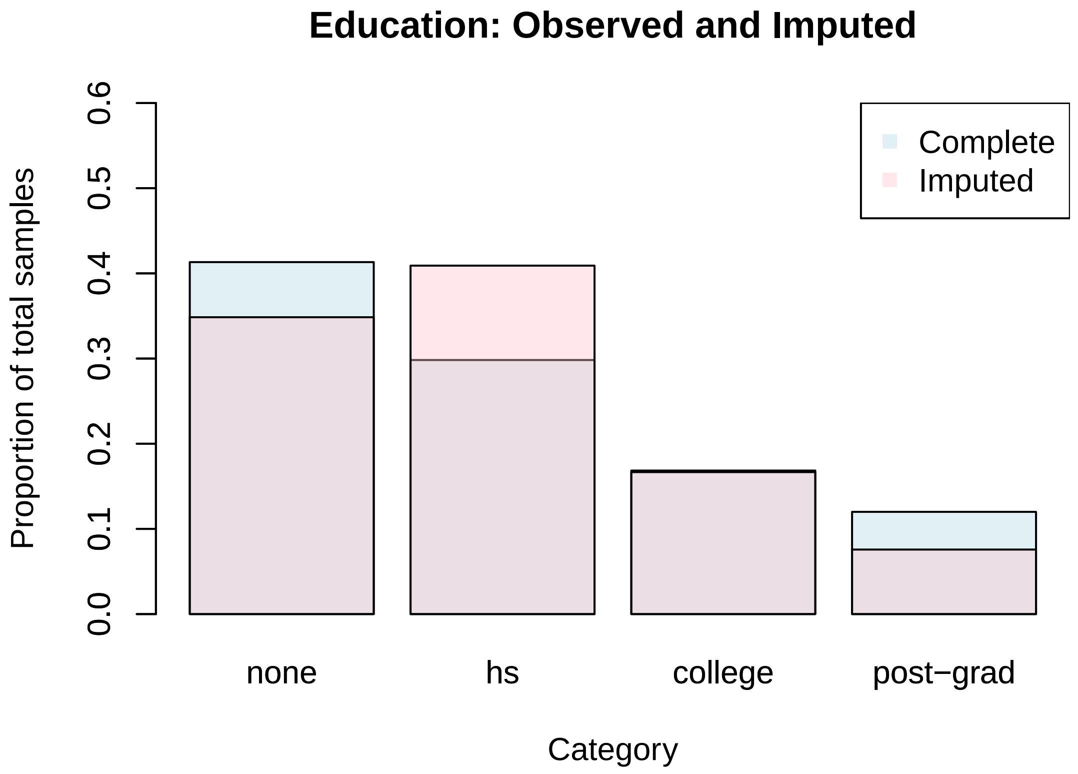
We can use logreg for BPMeds - since it is binary -, polyreg for education and linear regression for all other columns with missing

We use the package called mice van Buuren and Groothuis-Oudshoorn (2022) to imputed the data. Furthermore, since we do not want to use the test data to impute on itself we use the function `mice.reuse` from the package NADIA Borowski and Fic (2022).

```r
library(mice) # mice()
library(NADIA) # mice.reuse()

# 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 modelusing the training data
imp_mod.train = mice(d.train, m=1, printFlag = F, method = meth)
d.train.imp = complete(imp_mod.train)

# Use the imputation model trained on the training set on the test set to avoid data leakage
# Hva skjer med heartRate når vi bruker mice.reuse?
imp_mod.test = mice.reuse(imp_mod.train, d.test, printFlag = F)
d.test.imp = imp_mod.test$"1"

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



```
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$education)), col = c1, xlab = "Category", ylab = "Proportion
barplot(prop.table(table(d.train.imp$education[is.na(d.train$education)])), col = c2, add = TRUE)
legend("topright", legend=c("Complete", "Imputed"), cex=1, pch=15,col=c(c1,c2))
```



Model

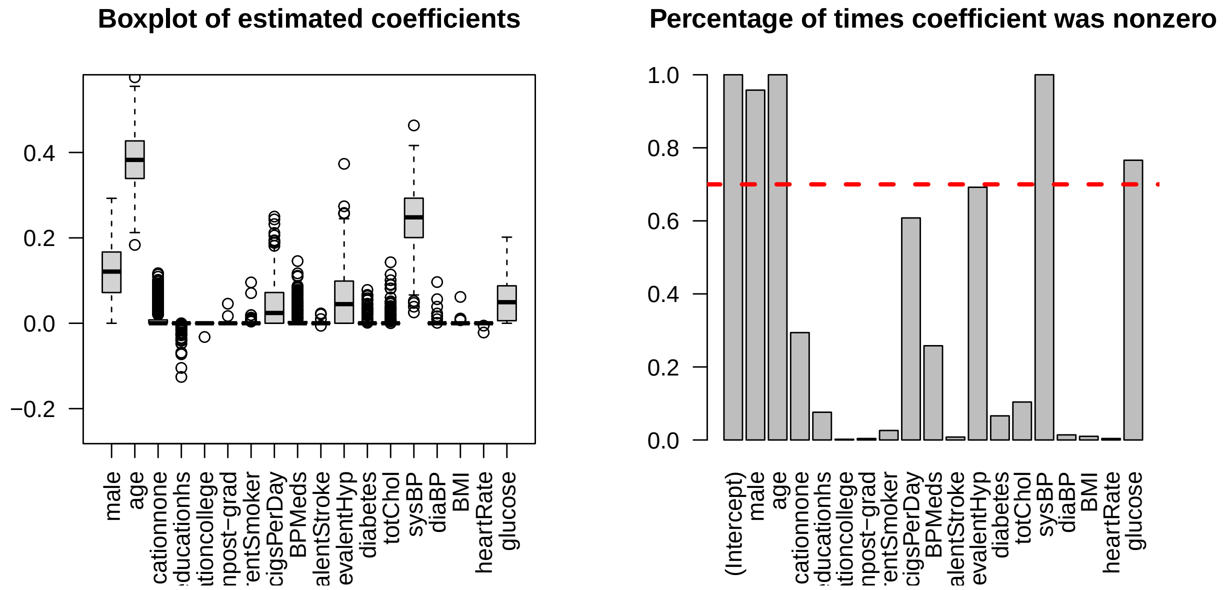
In this section we fit the same model on both the imputed and complete training data. The two data sets are indentical apart from the rows that contained `NA`'s. Since we want to do lasso, we first standardize the data. Secondly, we use bootstrap to calculate the non-zero coefficients multiple times.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

```r
set.seed(8701)
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

# 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 coeffic
barplot(apply(B.coef.count.complete, 2, sum)/B, las = 2, main = "Percentage of times coefficient was non
abline(h=0.7,col="red", lty=2, lwd=3)
```



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

After using lasso on 500 data sets the variables that have nonzero coefficients in at least 70% of the time, are (Intercept), male, age, sysBP, glucose. They are also indicated on the right figure above. We use the chosen covariates to fit a logistic regression model. To avoid overfitting, we use the independent test data to estimate the coefficients. This can be thought of as single split????

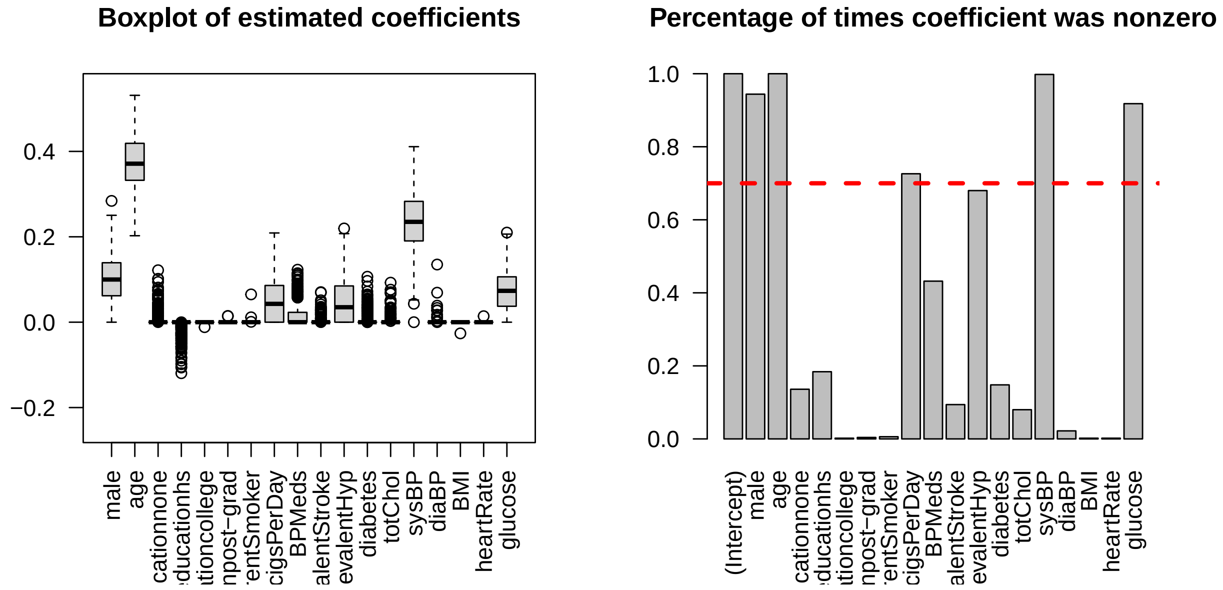
Something about Mette's comment on how we did inference.

Should we say something about the boxplot as well?

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 = bino
```

The Lasso on the imputed data chooses (Intercept), male, age, cigsPerDay, sysBP, glucoseas the non-zero covariates, here also 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. However, we question whether this might be a poor choice, as the training data were used to build the imputation model that were used to predict the missing values in the test set.

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 1020.37
```

```
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
## -1.5167  -0.6054  -0.4705  -0.3364   2.6711
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.547948   0.690551 -10.930  < 2e-16 ***
## age          0.056863   0.010543   5.393 6.91e-08 ***
## male         0.383116   0.172167   2.225 0.026064 *
## sysBP        0.014077   0.003617   3.892 9.93e-05 ***
## glucose      0.007674   0.002826   2.716 0.006616 **
## cigsPerDay   0.022526   0.006725   3.350 0.000809 ***
## ---
## 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: 1008.4  on 1265  degrees of freedom
## AIC: 1020.4
##
## 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) -8.929814820 -6.21947856
## age          0.036347183  0.07772072
## male         0.045498113  0.72121913
## sysBP        0.006992055  0.02119983
## glucose      0.002132940  0.01338914
## cigsPerDay   0.009279200  0.03569236
```

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 for curiosity we check also a model fitted using the imputed test set, but with the same covariates as chosen by 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
## -1.4898  -0.6082  -0.4753  -0.3571   2.5894
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.023624   0.661574 -10.617  < 2e-16 ***
## age          0.049999   0.010176   4.913 8.95e-07 ***
## male         0.552623   0.162610   3.398 0.000678 ***
## sysBP        0.014035   0.003600   3.899 9.65e-05 ***
## glucose      0.007277   0.002786   2.612 0.009000 **
## ---
## 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: 1019.3  on 1266  degrees of freedom
## AIC: 1029.3
##
## Number of Fisher Scoring iterations: 5
```

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

```
##                    2.5 %      97.5 %
## (Intercept) -8.345846741 -5.74915824
## age          0.030153437  0.07008605
## male         0.234840570  0.87303488
## sysBP        0.006987823  0.02112739
## glucose      0.001772278  0.01288639
```

```
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 1020.37             1029.35
```

```
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
```

Finish this train of thought

We did not use ROC-AUC to evaluate the model as both train and test data has been used to fit the final model. Hence, a third, independent data would be needed in order to run ROC-AUC calculations…

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.

Use the boxplot (sd not in 0 something) instead of cutoff percentage for times they were non-zero…?

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.

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.

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.