Statistical Programming Language: South African Heart Disease Study

Due on March 30th, 2018

Alla Petukhina

Lukas Bargel (572482), Adrian Rolf (572656) and Felix Vala (518231)

Table of Contents

List of Figures

List of Tables

1 Introduction

Dying from a heart disease is still the leading cause for death worldwide. 15,5% of deaths in the world in the year 2015 (see World Health Organization, 2017) can be explained by ischaemic heart disease. As this is even stronger the case for upper-middle and high-income countries (18,3% and 16,9%), measures to prevent illness of the heart should be a number one priority for health care providers in those countries. To be able to design preventive measures to avoid heart disease from occurring, one should identify the major risk factors for this illness.

This paper revisits a classical article from a medical journal. In 1983, a group of researchers around J. E. Rossouw investigated the occurrence of heart disease in South Africa and collected data on life circumstances they considered to be risk-factors (see Rossouw et al., 1983). The aim of this paper is to reproduce the results of a classical empirical paper using current statistical methods. Thereby, capabilities of using of the statistical programming language *R* will be demonstrated. *R* is a language and environment for statistical programming and graphics. It is an evolution of the statistical language *S*, which was originally developed at AT&T's research and development by John Chambers and colleagues. *R* is very extensible as its open source nature provides the possibilities for developers to release extension packages, of which a few will be presented in this paper (cf. R-Foundation, 2018).

The paper is organized as follows. Section 2 gives a theoretical introduction to the statistical methods used in this paper. Section 3 explores the data set by using descriptive statistics. In section 4, the practical implementation of the methods presented in section 2 begins with the Fisher's linear discriminant in 4.1. In section 4.2, the Logistic Regression will be performed with different models and these will be evaluated based on their out-of-sample-performance in section 4.3 by implementing the leave-one-out cross-validation. As the last part of the analysis, the observations are examined for single outliers which heavily influence the model by Cook's distance in section 4.4. Furthermore section 4.5 summarizes the results of the previous chapters and section 5 describes the practical conclusions of the statistical results.

2 Methods

In the case at hand – the prediction if heart disease occurs or not given some crucial indicators – we are dealing with a binary classification problem. There are several methods that can be used for such problems.

2.1 Fisher's Linear Discriminant

One convenient way is to use a linear discriminant or in particular, as we will do here, a Fisher's linear discriminant.

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote a data matrix and $c \in \{0, 1\}$ a class, known a priori, that divides the data into two subsets $\mathbf{X}_0$ and $\mathbf{X}_1$. The goal is now to find a separation rule that allows to allocate each observation – or new observations – into one of these subsets. The *best* separation rule is the one that minimizes the misclassification rate. Fisher's approach to this minimization problem is, following the notation of Duda et al. (2001), to find a vector $\mathbf{w}$ that projects each observation $\mathbf{x}_i$, $i = 1, \ldots, n$ onto a one-dimensional subspace $y_i = \mathbf{w}^T \mathbf{x}_i$, that maximizes the ratio

$$J(\mathbf{w}) = \frac{|\tilde{m}_0 - \tilde{m}_1|^2}{\tilde{s}_0^2 - \tilde{s}_1^2}, \tag{1}$$

where $\tilde{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathbf{X}_i} \mathbf{w}^T \mathbf{x}$ denotes the sample mean and $\tilde{s}_i^2 = \sum_{\mathbf{x} \in \mathbf{X}_i} (\mathbf{w}^T \mathbf{x} - \tilde{m}_i)^2$ the scatter matrix of the projection of the subset $\mathbf{X}_i$, $i = 0, 1$. That is, finding a rule that keeps the *within-group-variance* as small as possible while keeping the *between-group-variance* as large as possible at the same time. The rule $\mathbf{w}$ that maximizes equation 1 can be easily obtained by

$$\mathbf{w} = \mathbf{S}_w^{-1}(\mathbf{m}_0 - \mathbf{m}_1), \tag{2}$$

where

$$\mathbf{S}_w = \mathbf{S}_0 + \mathbf{S}_1, \tag{3}$$

with

$$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathbf{X}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \tag{4}$$

is the sum of the scatter matrices of the non-projected data. $\mathbf{w}$ is called the *Fisher's linear discriminant.*

The decision rule whether to allocate an observation to $\mathbf{X}_0$ or $\mathbf{X}_1$ now becomes a simple constant d that serves as a threshold. If we center the data first, this threshold is just 0. Thus, for each observation we finally get the discrimination rule:[1]

$$\mathbf{w}^T \mathbf{x}_i \geq 0. \tag{5}$$

2.2 Logistic Regression

Another approach to binary classification problems, that comes with few assumptions and a representation as a model that can be easily used for interpretations is *Logistic Regression.* Let again $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote a data matrix. The general idea of linear regression is to represent a *response variable* $\mathbf{y}$ in terms of a linear function of a set of *explanatory variables* $\mathbf{x}_1 \ldots \mathbf{x}_q$, $q \leq p$:

$$y_i = \beta_0 + \beta_1 x_{1i} + \ldots \beta_q x_{qi} + \epsilon_i, \tag{6}$$

where $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0,\, \sigma_\epsilon^2)$ is a stochastic error term. Taking the conditional mean of $\mathbf{y}$, the (zero mean) error term just drops out and we get, stated more conveniently in matrix notation:

$$\mathbb{E}[\mathbf{y} \,|\, \mathbf{X}] = \mathbf{X}\boldsymbol{\beta}. \tag{7}$$

An estimate for $\boldsymbol{\beta}$ can be obtained using Maximum Likelihood estimation and can be expressed analytically as $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. However, here we are assuming that $y_i \sim \mathcal{N}(\mathbf{x}_i^T \beta,\, \sigma_\epsilon^2)$. For our binary classification problem, i.e. where $y_i \in \{0, 1\}$, this is not a very reasonable assumption. A more general approach to regression models that allow for several other distributions for y_i are *Generalized Linear Models.*

The idea of generalized linear models is to use a link function ϕ to express the model

[1]Note that this problem can get quite more complex if non-equal piror probabilities and different distributions of the variables are involved. Thus, implicitly equal priors and a multivariate normal distribution are assumed here, making the calculation of $\mathbf{S}_w$ and spotting of the threshold computational simple.

as

$$\phi(\mathbb{E}[\mathbf{y} \mid \mathbf{X}]) = \mathbf{X}\boldsymbol{\beta}. \tag{8}$$

It can be shown that there are several distributions – from the *exponential family* – that can be represented by a link function ϕ and that in such cases Maximum Likelihood estimation can be used to obtain estimates of the parameters (see e.g. Spokoiny and Dickhaus, 2015). If we choose $\phi(x) = ln(\frac{x}{1-x})$ to be the *logit function* we obtain on the other hand

$$\mathbb{E}[\mathbf{y} \mid \mathbf{X}] = \phi^{-1}(\mathbf{X}\boldsymbol{\beta}) = \frac{e^{\mathbf{X}\boldsymbol{\beta}}}{1 + e^{\mathbf{X}\boldsymbol{\beta}}}, \tag{9}$$

where ϕ^{-1} is called the *logistic function.* This model is suitable to represent Bernoulli distributed response variables and is called a *Logistic Regression Model.* Estimates for $\boldsymbol{\beta}$ need to be obtained numerically. Fast algorithms for this purpose are for example the *Reweighted Least Squares Algorithm* (see e.g. Wasserman, 2013), which is the underlying algorithm in *R* for generalized linear models.

2.3 Leave-One-Out Cross-Validation

After fitting a model the question arises whether this model would still perform well on another dataset, sometimes referred to as the *out-of-sample performance.*[2] That is, to prevent over-fitting of the model to some particular dataset. But often, there is no other data at hand. One idea is then to split the data into a training and a test sample, the first to fit the model and the second to check its performance on the – from the view of the training sample – out-of-sample data. This can be repeated for different subsets and some measure over all the predictions can then be calculated, e.g. the *Mean Squared Error* or the misclassification rate. These measures can in turn be used as a benchmark for different models. This method is called *Cross-Validation.* We can choose the training sample to contain $n-1$ of the n observation and the testing sample the remaining one. Repeating this procedure for every single observation is called *Leave-one-out Cross-Validation.*

[2]In the following, for simplicity, when we use this term, we refer to it as a measure of the out-of-sample performance obtained with leave-one-out cross-validation.

2.4 Cook's Distance

The *Cook's Distance* is a method to detect influential observations in regression models. It was proposed by Cook (1977) and is defined as

$$D_i = \frac{\sum_{j=1}^{n} \left(\hat{y}_j - \hat{y}_j^{(i)}\right)^2}{p\ \widehat{\text{MSE}}}, \tag{10}$$

where $\hat{y}_j$ is the predicted value from our model for observation j, $\hat{y}_j^{(i)}$ the predicted value from our model for observation j excluding observation i and $\widehat{\text{MSE}} = \frac{1}{n-p}\sum_{i=1}^{n}(y_i - \hat{y})^2$ is an estimate of the mean squared error. As can be seen from equation (10), a high value of D_i compared to the values of the other variables indicates an influential observation i, since we would expect the deviations in the nominator to be approximately the same for all i if it doesn't matter whether we exclude it or not.

3 Descriptive Statistics

The following section deals with the discription of the data set (section 3.1) and the graphical exploration (section 3.2).

3.1 Description of Data Set

A postal campaign recruited 3357 white males and 3831 white females between the ages of 15 and 64 years as the population sample of the study. They represent 68% of the 1980 census population aged 15 – 64 years. Respondents completed a risk factor questionnaire and a London School of Hygiene questionnaire for chest pain by interview in a study centre set up in each community. The interviewers were dietitians who had been intensively trained in the standardized administration of the questionnaire. The questionnaire covered socio-economic items, smoking habits, a family history of heart disease, personal medical history and activity patterns. Topics like risk factor knowledge, attitudes and actions were covered as well. The respondents also completed a self-administered Bortner Short Rating Scale for coronary-prone (typea) behaviour. (see Rossouw et al., 1983). This set-up lead to the following variables:

- To find out about the subject's heart condition, a resting 12-lead electrocardiography was recorded in the recumbent position on a Hewlett-Packard Model No. 1516 Tape Terminal for later playback and coding according to the Minnesota criteria. The coding was done in two stages and at each stage two independent blind observer were used, with an arbitrator for disagreements. This procedure lead to the dummy variable chd, which describes if the subject suffers from a heart disease or not.

- The variable sbp measures blood pressure. It was measured after the subject had been seated for 5 minutes. Following the American Heart Association guidelines for measuring blood pressure, a standard 12,5 x 23-cm cuff connected to a mercury manometer was used. Readings were taken three times and the lowest reading was recorded. Three observers were used to get safe results. Subsequent analysis of the data failed to show evidence of interobserver variation or of end-digit preference.

- tobacco observes tobacco usage in kilogramm. It measures the total tobacco consumed in the subject's lifetime and is calculated as the average per day multiplied by the period of use.

- The variable typea measures psychosocial stress on the self-administered Bortner Short Rating Scale.

- famhist is a dummy variable which describes if a family member of the subject suffered from a heart disease.

- ldl measures cholesterol levels. ldl is the amount of low-density lipoprotein. It is measured in mmol/l by taking and chemically analyzing[3] blood samples of all participants.

[3] Boehringer CHOD-P AP enzymatic method and dextran sulphate-magnesium chloride precipitation (Rossouw et al., 1983)

- obesity measures the ration of weight to body-height of a person. This is commonly referred to as the so called body mass index of a person.

- adiposity measures the percentage of body fat of a person. While the most common measure for obesity would also be the body mass index, there are cases, where a high weight to height ratio is not caused by unhealthy body fat. Therefore, this measured variable gives us concrete information about the effect of excess body fat as a risk factor for heart disease.

- age measures the age of a person in years when the coronary heart disease occurs.

- The variable alcohol gives the information of individual alcohol consumption per year in liter.

All in all, the data set contains one response variable and nine potential risk factors for its occurrence. The following analyzed data set is a retrospective sample of 462 males, which was collected after some participants received blood pressure reduction treatment and other programs to reduce their risk factors.

3.2 Graphical Exploration

For a continuous workflow, we start by loading the necessary extension packages. The package dplyr will be used for data manipulation. ggplot2 and gridExtra are packages with focus on data visualization. The package corrplot provides us with the function corrplot() for a correlation matrix. As the last package, ElementStatLearn includes the data set we us, namely SAheart. To create a robust code, it makes sense to be prepared for different scenarios. Additionally, warning messages in *R* are often quite difficult to understand for non-professionals. Therefore, we start by checking if all necessary packages are installed. installed.packages() gives a matrix which contains all packages. "!" defines as a logical NOT. So, if a package is missing, we formulated an easy understandable message for the user additionally to installing the package itself. With the option character.only = TRUE the library() command interprets the passed argument as a string variable instead of searching for the word pkg in the library.

```r
## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)
```

Code 1: Snippet of Quantlet SAHeart_Q1_Data_Preparation

Furthermore, we take care of the scenario when the PDF-path (save in folder desktop) does not exist. To check, we use if() in combination with dir.exist(), which gives back the information if the specified PDF-path exists. If that is not the case, the user is again informed with an easy understandable message and will be informed that the PDFs of the graphics will be saved in the current working directory. The name of the folder will appear in the message, as we use the function getwd() to show the name of the folder. paste0 concatenates the strings without spaces in between.

```r
## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
 if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}
```

Code 2: Snippet of Quantlet SAHeart_Q1_Data_Preparation

To get a first overview, we use the function glm(), which does the same as str() but shows as much data as possible. This is like a transposed version of print: columns run down the page, and data runs across. This makes it possible to see every column in a data frame (Wickham et al., 2017). The function str() is included in the code for comparison and the function head() for seeing the data in a table format. We also get information about the variable types. For comparison we included here both commands as well as the function head() that gives the data in an ordinary table format, where the number of the first rows to show may be specified as argument.

Additionally, we get a table of summary statistics by using the function summary().

```
## Some overview stuff

glimpse(SAheart)
str(SAheart)

head(SAheart, 5)

summary(SAheart)
```

Code 3: Snippet of Quantlet SAHeart_Q1_Data_Preparation

summary() is a useful generic function, which shows the results of various model fitting functions depending on the class of the object. As SAheart is a data.frame, we get a set of typical location parameters per variable (see table 1).

	Minimum	1st Quantile	Median	Mean	3rd Quantile	Maximum
sbp	101.0	124.0	134.0	138.3	148.00	218.0
tobacco	0.000	0.053	2.00	3.640	5.50	31.20
ldl	0.980	3.280	4.340	4.74	5.790	15.33
adiposity	6.740	19.77	26.11	25.41	31.23	42.49
typea	13.00	47.00	53.00	53.1	60.0	78.00
obesity	14.70	22.98	25.80	26.04	28.50	46.58
alcohol	0.000	0.510	7.510	17.04	23.89	147.2
age	15.00	31.00	45.00	42.82	55.00	64.00
famhist	Absent: 270	Present: 192				

Table 1: The summary statistics of the complete data set regarding the risk factors of coronary heart diseases (see related Quantlet SAHeart_Q1_Data_Preparation).

For example, we can immediately see that the age of the participants ranges from 15 to 64 years and there are less participants that have a family history of heart disease than have not. As *R* handles famhist as a factor variable, we have to rewrite it as an ordinary numerical variable, in order to calculate the correlation matrix using the corr() function, which is not able to handle factor variables. Factor variables are the way categorical variables are handled in *R* in various contexts and there are a lot of commands that can handle this variable type very utilitarian – including functions for regression models for example. On the other hand, there are also quite a lot of functions that cannot. Hence, it is advisable to convert to factor variables on the fly only when needed using the as.factor() function. Therefore, we decode as.factor as a dummy variable with 0 for absent and 1 for present. Technically, we refer to the underlying numerical values with the function as.numeric(). With the function head() for the first ten values we can observe that present is defined as 2 and absent as 1. We convert the factor variable to a numerical variable with as.numeric() and redefine absent as 0 and present as 1.

```r
## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())


## famhist
head(SAheart$famhist, 10)
head(as.numeric(SAheart$famhist), 10)
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1
```

Code 4: Snippet of Quantlet SAHeart_Q1_Data_Preparation

We start our exploratory analysis by creating a correlation matrix, which is a useful tool to get a systematic overview over correlations of the variables at a glance. From there, we get first intentions, which variables we should include in our model. We program the design of the matrix in such a way that intensity of color and shape of the graph show the strength and direction of the linear correlation between each of the variables. Technically, we therefore create a correlation matrix with cor() and use the select() function to put our response variable chd at the top. ColorRampPalette() gives us a vector of chosen coulors, we use in the function corrplot.mixed(). This function gives us various options for the design of the matrix. We use upper and lower to give us shapes and colors in the upper diagonal field and numbers in the lower diagonal field.

```r
## Correlation matrix (using package corrplot)
pdf(file = paste0(PDFpath,"/Corrplot.pdf"), width = 7, height = 8)
corr_matrix <- cor(select(SAheart, chd, everything(), -id))
colset <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot.mixed(corr_matrix,
               upper = "ellipse",
               lower = "number",
               number.cex = .7,
               upper.col = colset(10),
               lower.col = "black",
               tl.col = "black",
               tl.pos = "lt")
dev.off()
```

Code 5: Snippet of Quantlet SAHeart_Q2_Descriptive_Statistics

The most striking finding (see figure 1) is that the further analysis should focus on the variables sbp, tobacco, age, famhist, ldl and adiposity as only they show noteworthy correlations with our explanandum chd. Furthermore, we observe a significant correlation between some of the explanatory variables. adiposity and obesity may be a result of a similar nutrition, so the resulting variables are strongly correlated. As the last obvious point, obesity and adiposity are correlated with age.

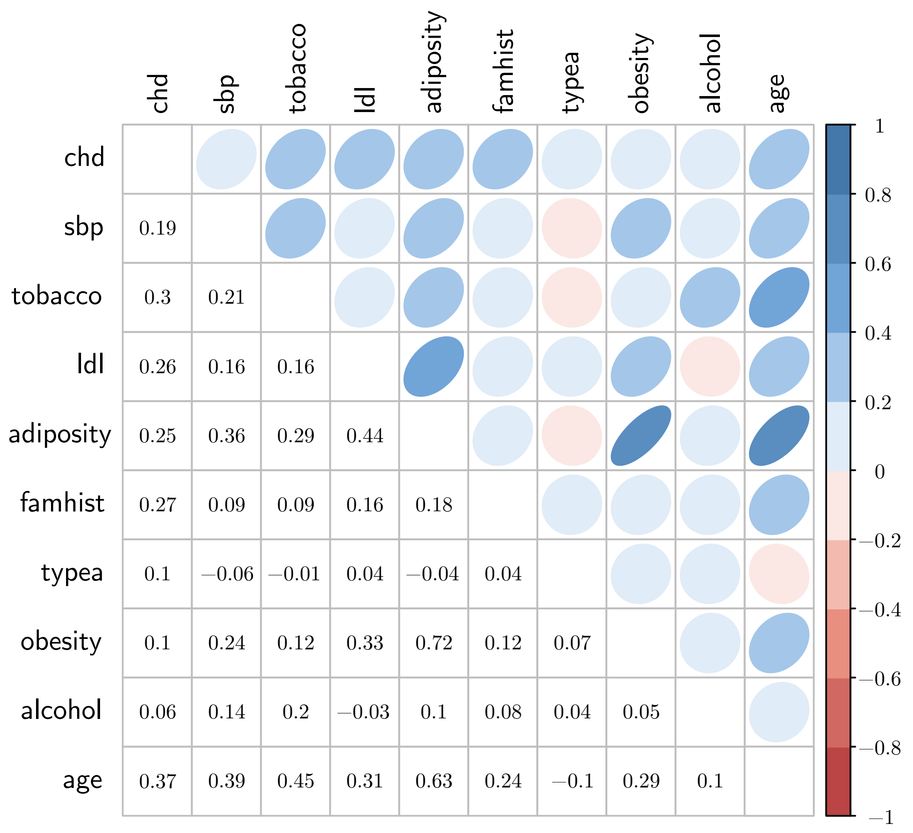

Figure 1: Correlation matrix of the different risk factors of coronary heart disease (see related Quantlet SAHeart_Q2_Descriptive_Statistics).

To further examine these first results, we program a number of boxplots. As we want to look at the distribution of the variables over 0 and 1 in chd we concentrate only on the five variables with high correlations as we don't expect distinguishable differences between the boxplots for 0 and 1 for the variables with low correlations. The Boxplots are created with the self-written function multiBP, that takes a list of variables that shall be plotted over chd as input. Optionally, a path can be specified to save the graphic in pdf format to the local machine. We use the function lapply to perform the generation of the plots on each of the variables contained in the input list. lapply in turn returns a list of specific ggplot graphic objects. ggplot is a powerful toolkit to handle and generate graphics within in *R* from data. Besides of specifying the underlying dataset and the variables to be used – where we convert chd on the fly to a factor variable – ggplot() takes various options and so called *layers* that can be connected via a "+" symbol. The different layers make it easily possible to merge different illustration of the same data into one graphic. Here we only use one layer for the boxplots. Also, the appearance of the graphic can be edited in great detail or just templates - like theme_bw() – can be used, as we do here. Since the number of boxplots that will be saved to plot.list can vary, we account for the arrangement of the single boxplots in one graphic. If there are more than three boxplots, the number of columns is restricted to three, so that the grid.arrange() function will successively push the fourth and seventh into the next rows. Otherwise all boxplots would be displayed in one row and therefore heavily shrinked in their width. grid.arrange() as well as ggsave are functions specially designed to handle ggplot graphic objects.

```
## Function to show Boxplots of covariates over chd
multiBP <- function(varlist, PDFpath = NULL){
  plot.list <- lapply(varlist, function(ivar){
    ggplot(data = SAheart, mapping = aes(x = as.factor(chd), y = get(ivar))) +
      geom_boxplot(stat = "boxplot", position = "dodge") +
      theme_bw() +
      xlab("chd") +
      ylab(ivar)
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3
  }

  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)

  if (!is.null(PDFpath)){
    ggsave(filename = "/BPplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }
}


 varlist <- c("sbp", "tobacco", "age", "ldl", "adiposity")
multiBP(varlist, PDFpath)
```

Code 6: Snippet of Quantlet SAHeart_Q2_Descriptive_Statistics

All boxplots solidify our assumptions (see figure 2), that the five variables might be important contributors for our model. The boundaries of all quantiles lie higher for observations where the participants of the study suffered from heart disease. Especially age and tobacco show a higher median at chd $= 1$ than where the 75% quantile of the values with chd $= 0$ lies. Boxplots also give first signs for outliers. Observations, which are further away from the box than 1,5 times the interquartile range, are shown as dots. We can observe an especially noteworthy outlier in the boxplot graphic belonging to the variable age as it is the only outlier to the bottom. As we take a closer look at the observation, we can see that the person has quite – medically spoken – unobtrusive values for all risk factors and suffers from heart disease nevertheless. We keep that in mind for our model selection later. Furthermore, we observe relatively strong outliers on sbp and ldl. Despite the thorough procedure of measurements, it is possible, that these are measurement errors since the occurrence of such high values in the human body is unlikely.

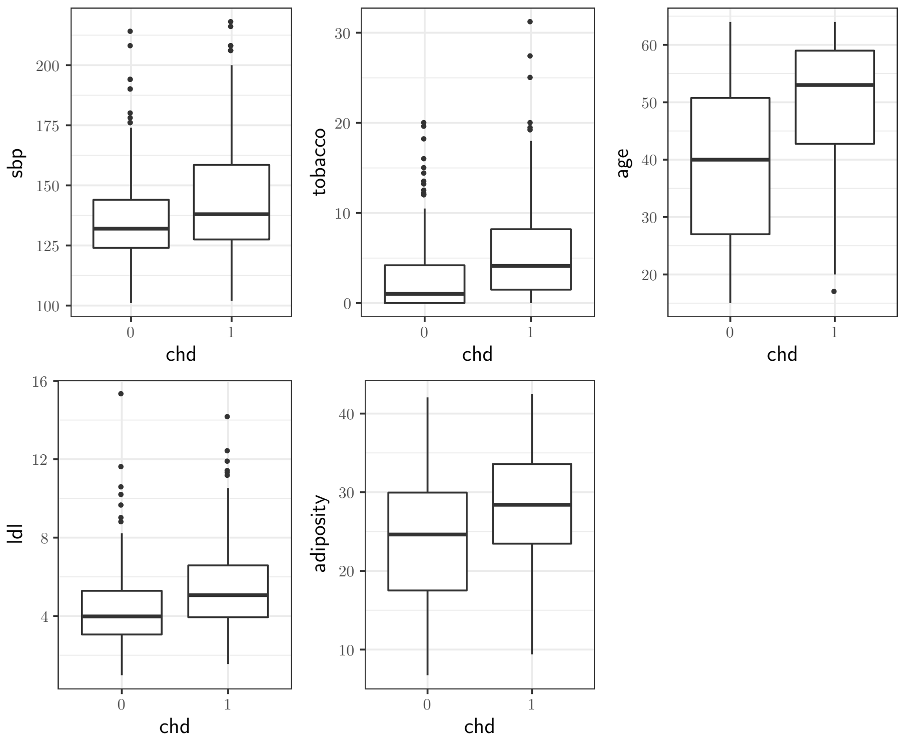

Figure 2: Boxplots for different risk factors over occurence of coronary heart disease (see related Quantlet SAHeart_Q2_Descriptive_Statistics).

As the next part of our analysis, we want to observe the behavior of selected outliers. From the boxplots we know, that there is a very low value of age at chd = 1. So the scenario we look at in the following are very young participants (defined as age <= 20), who suffer from heart disease. To get the IDs of these observations, we let R show us the values of the variable ID, but only those with the restrictions that the variable age is smaller or equal 20 and chd is equal to 1. With the "&" sign we define that both conditions have to be met.

```r
## Suspicious outlier
SAheart$id[SAheart$age <= 20 & SAheart$chd == 1]

subset(SAheart, select = varlist, id == 261)
subset(SAheart, select = varlist, id == 21)
summary(subset(SAheart, select = varlist, chd == 1))
```

Code 7: Snippet of Quantlet SAHeart_Q2_Descriptive_Statistics

As a result, we get to know that the conditions are met by the observations with IDs 21 and 261. To further examine these two observations, we use the function subset() to show the values (see table 2) of these specific variables.

	261	21
sbp	118	106
tobacco	0.00	1.61
age	17	20
ldl	2.39	1.74
adiposity	12.13	12.32

Table 2: Selected outliers and their oberservations of the five most influential risk factors for coronary heart disease (see related Quantlet SAHeart_Q2_Descriptive_Statistics).

For comparison, we create some summary statistics (see table 3) with the function summary() as before, but this time only for the five most influencial risk factors and for observations where chd = 1. A first result is that our two selected observations have unusually low values for nearly all variables.

	Minimum	1st Quantile	Median	Mean	3rd Quantile	Maximum
sbp	102.0	127.5	138.0	143.7	158.5	218.00
tobacco	0.000	1.500	4.130	5.525	8.200	31.200
age	17.00	42.75	53.00	50.29	59.00	64.000
ldl	1.550	3.940	5.065	5.488	6.582	14.160
adiposity	9.390	23.46	28.41	28.12	33.59	42.490

Table 3: The summary statistics of the five most influencial risk factors of coronary heart disease for chd = 1 (see related Quantlet SAHeart_Q2_Descriptive_Statistics).

To visualize these results, we create one jitterplot per variable, where the two selected outliers are highlighted in color. That is reasonable since we can see a first glance of the behavior of the highlighted observations in comparison to all others on the selected variables. For this purpose, serves the function multiJP, which takes varlist as an obligatory argument. Optionally, there may be specified a list of observations via IDs to be highlighted, and if they shall be labeled with their IDs. And again, it is possible to specify a path to save the graphic in pdf format to the local computer. First, the function checks if the Boolean parameter label was set to TRUE. If so, the varlist is expanded by an additional entry, indicating that a legend should be set. Like above, this varlist is passed to an lapply() function that will return a list containing the single jitterplots. To obtain a legend that gives the IDs of the highlighted observations we use a workaround here – that is why the corresponding variable is called hackdata – since a proper legend can't easily be created with the standard function in this special case here.

```r
multiJP <- function(varlist, obslist = 0, PDFpath = NULL, label = FALSE){


  if(label){
    varlist <- c(varlist, "legend")
  }

  color.list <- c("firebrick", "dodgerblue", "forestgreen",
                  "orangered", "violet", "yellow3")

  plot.list <- lapply(varlist, function(ivar){
    if(ivar == "legend"){
      hackdata <- data.frame(id = obslist,
                             position = seq(1,length(obslist)),
                             xvalue = 0.11)
      legend <- ggplot(data = hackdata, mapping = aes(x = xvalue, y = position)) +
        geom_point(colour = color.list[1:length(obslist)], shape = 17, size = 3) +
        theme_classic() +
        theme(axis.text = element_blank(), axis.ticks = element_blank(),
              axis.line = element_blank()) +
        labs(x = "", y = "") +
        scale_y_continuous(expand = c(0, 0), limits = c(0, max(hackdata$position)+1)) +
        scale_x_continuous(limits = c(0.1, 1)) +
        geom_text(data = hackdata, aes(label = id),
                  colour = color.list[1:length(obslist)], size = 3, hjust = -1)

    }else{

      ggplot(data = SAheart[!(SAheart$id %in% obslist),],
             mapping = aes(x = as.factor(chd), y = get(ivar))) +
        geom_jitter(shape = 1) +
        theme_bw() +
        xlab("chd") +
        ylab(ivar) +
        geom_jitter(data = SAheart[SAheart$id %in% obslist,],
                    colour = color.list[1:length(obslist)], shape = 17, size = 3) # +
    }
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3 }

  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)

  if(!is.null(PDFpath)){
    ggsave(filename = "/Jitterplot_outliers.pdf", path = PDFpath, device = "pdf", plot = final.plot)}}
```

Code 8: Snippet of Quantlet SAHeart_Q2_Descriptive_Statistics

The idea is to add an additional graphic that is build such that it serves as a legend. This is obtained by removing all elements like axes or gridlines and only displaying the highlighted observations in the corresponding color and their ids attached to them as labels. All necessary information is saved to hackdata as a data.frame. That is, the IDs as well as the positions of the symbols – depending on the number of observation – that shall appear in the legend. While scheduling the variables within the lapply()-procedure we check at each step if the last entry *legend* is reached. If not, a jitterplot for the current variable is created similar as the boxplots are created within the mulitBP() function described above. If instead the entry *legend* is reached

– given of course it was specified – the legend graphic will be build. This way lapply() returns a full list containing the jitterplot as well as the legend graphic. This is a crucial point here. Once the list was returned it is not possible to add additional graphics here since this list is already stored as a specific ggplot object ready to be processed further by specific functions, namely grid.arrange and ggsave() in our case. To use the function for our concrete problem, we define varlist as a vector of all variables, which we determined as relevant in the correlation matrix and obslist as a vector of the two selected outliers with IDs 21 and 261.

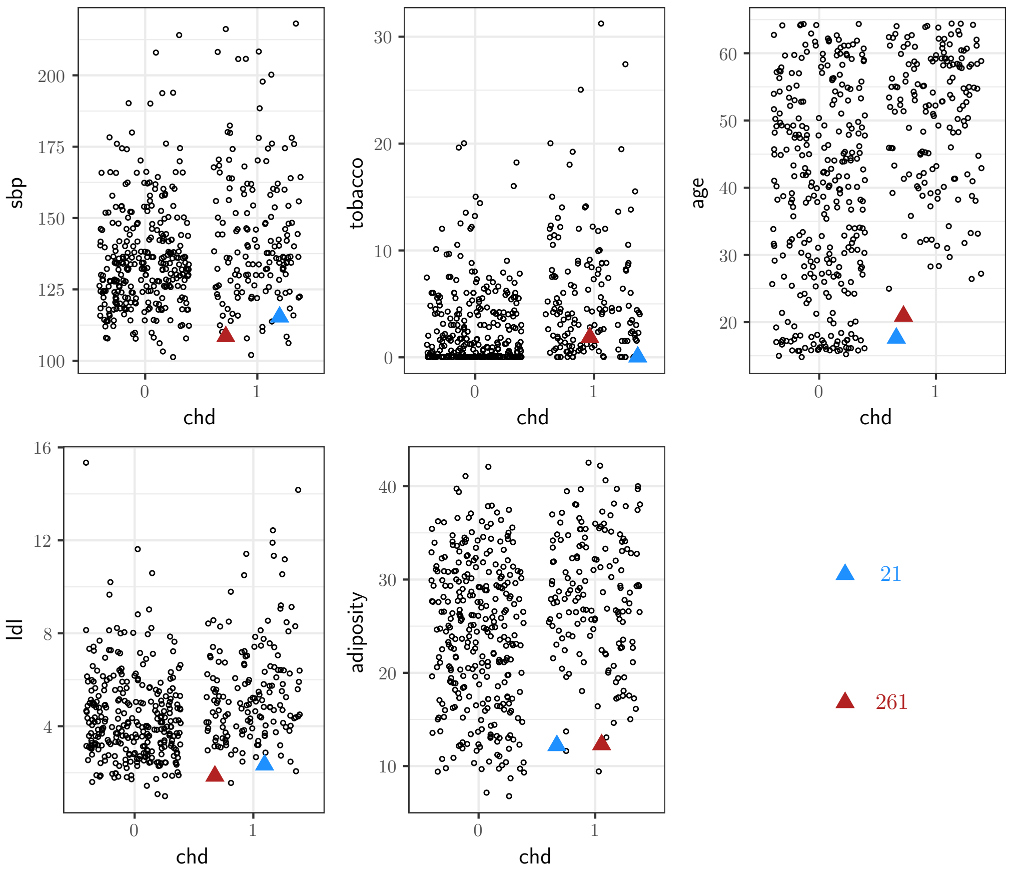

Figure 3: Jitterplot of selected outliers and the five most influencial risk factors of coronary heart disease (see related Quantlet SAHeart_Q2_Descriptive_Statistics).

In the resulting graphic (see figure 3) we can observe that the two selected outliers have very low values on all variables. We keep that in mind for our model selection later. Very young persons who nevertheless suffering from heart disease are the most obvious possibility for outliers, but not the only one. To generally check for further conspicuous cases, we search for observations, which have very low values throughout all variables and anyhow suffer from heart disease (We check for low values, since they stand for a healthier lifestyle or body condition in all variables, so one wouldn't expect the occurrence of heart disease). Hence, we filter for cases where chd = 1 and all other variables lie in the 25% quantile. We let *R* give us the IDs of these observations and can see, that this scenario applies to none of the observations. Hence, in this case no further graphical examination is needed. However, we will make use of this function again in section 4.4.

4 Inferential Statistics

This section begins with the practical *R*-implementation of the methods presented in section 2 by coding the Fisher's linear discriminant in 4.1. In section 4.2, the Logistic Regression will be performed with different models and these will be evaluated based on their out-of-sample-performance in section 4.3 by implementing the leave-one-out cross-validation. As the last part of the analysis, the observations are examined for single outliers which heavily influence the model by Cook's distance in section 4.4. Furthermore section 4.5 summarizes the results.

4.1 Fisher's Linear Discriminant

In this section we describe the implementation of the function for the *Fisher's linear discriminant.* First, we calculate the Fisher's linear discriminant $\mathbf{w}$ as described in section 2.1. Then, we apply the decision rule from equation (5) to the data and calculate the prediction accuracy as $1 -$ *missclassification rate*, i.e. the proportion of correct predictions. Further, we will plot the densities of the projected data according to the two classes to visualize how good this decision rule is able to difference between these two classes.

The function fisher takes three arguments. First, X, which can be a matrix or a data.frame object of dimension $(n \times q)$, that contains the covariates, i.e. the set of variables that shall be used to classify into one or the other class. Second, c, a binary vector of dimension $n \times 1$, containing the respective class for every observation. Note, that both arguments are obligatory since no default is set. And, third, an optional parameter PDFpath that can, again, be used to directly save the graphical output – in this case the densities of the projected data given the classes – to the local computer.

In line 54 we use the as.matrix() function to convert the input into a matrix in case it is given as a data.frame object, since the operations in the following calculations require matrix objects. We then split the matrix into two matrices according to their classes using simple index notation. Leaving the right side of the comma blank means that no condition for the single columns is stated and therefore refers to all columns. The command nrow() returns the number of rows. In lines 256 and 257 we calculate the mean for each column of the two matrices and convert them to a vector to get a plain numeric vector without additional information like the variable names saved to this object.

```
51 ## Function for Fisher Linear Discriminant
52 fisher <- function(X, c, PDFpath = NULL){
53
54   X <- as.matrix(X)
55
56   X0 <- X[c == 0,]
57   X1 <- X[c == 1,]
58
59   n0 <- nrow(X0)
60   n1 <- nrow(X1)
61
62   m0 <- as.vector(colMeans(X0))
63   m1 <- as.vector(colMeans(X1))
64
65   M0 <- as.vector(rep(1, n0)) %*% t(m0)
66   M1 <- as.vector(rep(1, n1)) %*% t(m1)
67
68   X0_c <- X0 - M0
69   X1_c <- X1 - M1
70
71   S0 <- t(X0_c) %*% X0_c
72   S1 <- t(X1_c) %*% X1_c
73
74   S_w <- S0 + S1
75
76   S_w_inv <- solve(S_w)
77
78   w <- S_w_inv %*% (m0 - m1)
```

Code 9: Snippet of Quantlet SAHeart_Q3_Fisher_Linear_Discriminant

This prevents errors during matrix operations. Not converting it sometimes causes the error, that *R* doesn't recognize it as a proper numeric vector. However, the results are two $(q \times 1)$ vectors.

In the next step we want to center the matrices $\mathbf{X}_i$ using the respective mean vectors $\mathbf{m}_i$, $i = 0, 1$. What is mathematically easily expressed as $\mathbf{X}_i - \mathbf{m}_i$ needs a little more work in *R*. *R* can't handle operations like $\mathbf{Z}_{(m \times n)} - \mathbf{z}_{(m \times 1)}$. The operands need to have the same dimension. For this purpose we want to create matrices that contain n times the q means for each variable in the columns. This can be achieved by

$\mathbf{1}_{(n_i \times 1)}\mathbf{m}_i^T$, where $\mathbf{1}_{(n_i \times 1)}$ is a vector of ones. $\mathbf{1}_{(n_i \times 1)}$ is created using the function rep() that returns a sequence of ones of length n_i in this case. This sequence is then converted to a vector so R can easily handle it during matrix operations.
The %*% denotes the matrix multiplication operation and the function t() transposes a matrix. With these build-in operations of *R* we can calculate the $\mathbf{M}_i$ and finally use them to center the matrices $\mathbf{X}_i$ using simple subtraction.
The same goal could have of course been achieved by using a loop running over each row of $\mathbf{X}_i$ subtracting $\mathbf{m}_i$. But the larger the covariates matrix $\mathbf{X}$ the more computational effort is needed to run over every single row. Matrix operations are very efficient implemented in the most programming languages, including *R*. Hence, whenever possible, programmed algorithms should use matrix operations instead of iterations to avoid time consuming calculations.
Using the centered, data we calculate the sum of the scatter matrices as described in equation (3) and (4). To get the inverse of $\mathbf{S}$ we use the function solve(a, b), where a and b are two arguments. The purpose of this function is to solve a system of equations that is given in terms of matrices and vectors $\mathbf{Ax} = \mathbf{b}$. If the argument b is not specified an identity matrix $\mathbf{I}$ of the respective dimension according to $\mathbf{A}$ is assumed as default. Thus, the equation becomes $\mathbf{Ax} = \mathbf{I}$ and the solution to this equation is $\mathbf{A}^{-1}$. Therefore, this function can easily be used to obtain the inverse of a matrix – given it is non-singular, of course. Finally, we have all components together to calculate the Fisher's linear discriminant $\mathbf{w}$.

```
m <- as.vector(colMeans(X))
M <- as.vector(rep(1, nrow(X))) %*% t(m)
 X_c <- X - M

 z <- t(w) %*% t(X_c) < 0

 accuracy <- sum(t(z) == c) / length(z)
```

Code 10: Snippet of Quantlet SAHeart_Q3_Fisher_Linear_Discriminant

Afterwards we center the whole covariates matrix X as above. The projection of the centered data is then used for the decision rule described in equation (5). To apply the decision rule on the whole covariates matrix at once instead for every single observation, $\mathbf{X}_c^T$ is used – since we achieve the same goal by $\mathbf{w}^T\mathbf{X}_c^T$ as by calculating $\mathbf{w}^T\mathbf{x}_i$ n-times. If the value of the projected data is lower than zero the $(n \times 1)$ vector $\mathbf{z}$ will contain TRUE for this row and FALSE otherwise. Thus, $\mathbf{z}$ consists of the predictions to which class an observation is allocated by the decision rule, where TRUE refers to class 1. That is in our case the prediction that an observation suffers from coronary heart disease.

Furthermore, the prediction accuracy is calculated. $\mathbf{z}$ is transposed here so that R can check for equality on each row of $\mathbf{z}$ and $\mathbf{c}$ in one step. This comparison is possible here since the 1 and 0 entries of $\mathbf{c}$ are automatically handled as TRUE and FALSE, respectively. Each comparison in turn yields TRUE or FALSE such that summing this up is just like counting the number of TRUE-entries. Dividing by the total number of rows, we finally get the prediction accuracy. Note, that we use length() here since it returns n in both cases, whether the input is a $(n \times 1)$ or an $(1 \times n)$ vector. Otherwise, depending on the dimension in which the vector is currently given within R, one has to check whether nrow() or ncol() is appropriate. Thus, in such cases length is the better choice to avoid mistakes that can easily been overseen at first glance.

```
88  Y0 <- X0 %*% w
89  Y1 <- X1 %*% w
90
91  Y <- rbind(cbind(Y0, rep(0,nrow(Y0))),
92             cbind(Y1, rep(1,nrow(Y1))))
93  Y <- as.data.frame(Y)
94  colnames(Y) <- c("Y","c")
```

Code 11: Snippet of Quantlet SAHeart_Q3_Fisher_Linear_Discriminant

In order to plot the densities of the projected data according to the classes, we first need to calculate the respective vectors containing these data. This is done in lines 88 to 89. For each of the two vectors an additional column is added containing the respective class using cbind(), a command to easily bind columns of vectors and/or matrices together. Again, we use the rep() command to create a vector of ones or zeros, respectively, of the right length on the fly. After that, these two matrices are binded again, this time horizontally using rbind(), yielding in the matrix $\mathbf{Y}$ containing the projection of every observation and the respective class. Furthermore, we need to convert $\mathbf{Y}$ into a data.frame object, since the following ggplot() command can only handle this type of objects. At last, we change the column names of $\mathbf{Y}$ in order to easily refer to its columns.

```
  final.plot <- ggplot(Y, aes(Y, fill = as.factor(c))) + geom_density(alpha = 0.2) +
    theme_classic() +
    labs(x = "", y = "") +
    theme(axis.text.x = element_blank(), axis.ticks.x = element_blank(),
          axis.text.y = element_blank(), axis.ticks.y = element_blank(),
          axis.line.y = element_blank()) +
      scale_fill_manual(values = c("firebrick", "dodgerblue"),
                        name = "chd", labels = c("no", "yes"))

  plot(final.plot)

  if(!is.null(PDFpath)){
    ggsave(filename = "/Densitiesplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }

  return(accuracy)
}


X <- select(SAheart, -id, -chd)
c <- SAheart$chd

fisher(X=X, c=c, PDFpath)
```

Code 12: Snippet of Quantlet SAHeart_Q3_Fisher_Linear_Discriminant

Furthermore, we create the plot of the densities of the projected data according to the two classes. The basic parameters of ggplot() have already been described in section 3.2. Here, it can be seen that using **Y** as a data.frame that contains the projected data as well as the respective classes makes it easy to use and the code easy to read. With geom_density() we define the type of the graph to be a density in this case. That is, Kernel density estimates are calculated for the given data, where a gaussian kernel is used by default. The parameter alpha $= 0.2$ makes the areas of the distributions transparent, so we can also see the overlapping areas.

The additional parameters specified within the theme() command change the look of the graphic, in particular removing the ticks of the axis as well as the labels and the line of the y axis. The specific values are not of interest here since we are only interested in how good the two densities are separated from each other. If they had no overlapping areas that would be the best result since then we could perfectly allocate any observation to the right class with a probability of zero for misclassification. But this is usually not the case.

With scale_fill_manual the color of the areas of the density is specified and at the same time the appearance and labels of the legend is configured. Finally, the graphic is plotted and, as already described in section 3.2, saved to the local machine if the PDFpath argument was specified. The prediction accuracy, i.e. a single value, is returned by the fisher function.

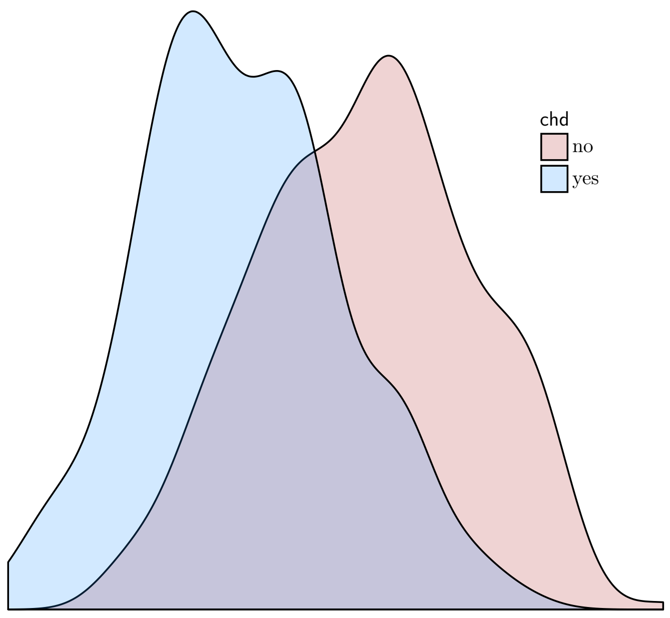

Figure 4: Densities of the data according to the two classes of chd after projecting it onto an one-dimensional subspace using the Fisher's linear discriminant (see related Quantlet SAHeart_Q3_Fisher_Linear_Discriminant).

After selecting all variables, except of chd and id, as the covariates matrix $\mathbf{X}$ and chd as the class vector $\mathbf{c}$, we run the fisher function. A prediction accuracy of ca. 0.695 is returned. That is a relatively satisfying result for such a case. At least it is better than a random classification of 0.5, but has still a misclassification rate of over 30%. The reason for that can be seen in Figure 4. The two densities have quite a large overlapping area. This means that in this cases we decide for the class that has a higher probability but there is still non-zero or even only a marginally less high probability to belong to the other class. Thus, there are a lot of cases where this decision rule might allocate an observation to the wrong class.
The Fisher's linear discriminant method has some weaknesses. We assume implicitly that every variable from the covariates matrix is normal distributed and this is of course not necessarily the case and especially for the binary variable famhist this assumption is obviously wrong. Anyway, Figure 4 already gives us a clue that we might not find a decision rule that can allocate the observations with a misclassification rate lower than 20% or even 10% at all. However, we can still try to improve the classification. In the next section we will use logistic regression models to formulate a decision rule. Logistic regression needs few assumptions – e.g. no

assumptions about the distributions of the covariates – and is known to be a robust method for binary response variables (Wasserman, 2013).

4.2 Logistic Regression

In the following we will use logistic regression to examine possible models. In line with the methodical section 2 we are using the generalized linear model function glm() from the package stats loaded in *R* by default. This function takes a regression model as an obligatory argument. The syntax is *dependent var : independent var1 + independent var 2 + .* Also, the underlying dataset has to be specified if the variable path is not provided in detail – i.e. like *somedataset$somevar* – or the respective dataset hasn't been attached to the R environment. Furthermore, we have to set the family parameter to indicate which specific case of a generalized model we are dealing with. In our case we have to set a binomial family and additionally the link to logit, see section 2.2 for more details (note that the Bernoulli distribution is a special case of the Binomial distribution, so this refers to the same distribution here). Again, we use the summary() command to obtain the regression outputs. As already mentioned in section 3.2, it can handle various models and provide suitable outputs.

```r
## M0: (naive) Full model
m0 <- glm(chd ~ ., family = binomial(link = "logit"),
          data = select(SAheart, -id))
summary(m0)


## M1: Wihtout sbp, alcohol; and exclude either adiposity or obesity
m1.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea + adiposity,
            family = binomial(link = "logit"), data = SAheart)
summary(m1.1)

m1.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea + obesity,
            family = binomial(link = "logit"), data = SAheart)
summary(m1.2)



## M2: Exclude sbp, alcohol, adiposity, obesity
m2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
          family = binomial(link = "logit"), data = SAheart)
summary(m2)


## M3: For sake of simplicity of models we try to exclude
## typea, since effect seems to be still very low
m3 <- glm(chd ~ tobacco + age + famhist + ldl,
          family = binomial(link = "logit"), data = SAheart)
summary(m3)
```

Code 13: Snippet of Quantlet SAHeart_Q4_Logistic_Regression

Despite the fact that we already received first hints on influential variables from the graphical exploration, we start with a naive full regression model m0, i.e. including

all variables. Thus, we possibly find some effects that get visible while controlling for other variables, since plain correlations do not account for that. A full model can easily be obtained using the single point shortcut, which serves as a placeholder for all variables of the specified dataset, except of the dependent variable, in this case. The output (see table 9) shows that sbp and alcohol have quite high p-values. Therefore, they are excluded in the next models.
We already know from theory as well as from figure 1 that obesity and adiposity are highly correlated. Hence, we perform two models m1.1 and m1.2 excluding obesity or adiposity, respectively. The outputs (for m1.1 see table 10 and for m1.2 see table 11) show that in both models the respective variable has a high p-values. Therefore, we exclude both variables and obtain the model m2, where all explanatory variables are significant at least at a 0.1% significance level. With regard to the relatively low correlation of typea with chd (see Figure 1), it was reasonable to exclude typea from m3. Nevertheless, the AIC of m3 (see table 12) is worse than the AIC of m2 and the estimated coecient of age in m3 is slightly weaker. Interestingly, typea has a significant effect, what we would not have expected from the graphical exploratory analysis. For sake of simplicity we exclude typea in model m3. This increases the AIC by more than 7 points compared to model m2.

```
## typea: mediator effect via age

mean(SAheart$chd[SAheart$typea >= median(SAheart$typea)])
mean(SAheart$chd[SAheart$typea < median(SAheart$typea)])

multiBP("typea")

obslist <- (SAheart %>%
              filter(chd == 1 & typea < 30) %>%
              select(id))$id

varlist <- c("sbp", "tobacco", "age", "ldl", "typea", "famhist")
multiJP(varlist, obslist, label = TRUE)


summary(glm(chd ~ typea, family = binomial(link = "logit"), data = SAheart))
summary(glm(chd ~ typea + age, family = binomial(link = "logit"), data = SAheart))
```

Code 14: Snippet of Quantlet SAHeart_Q4_Logistic_Regression

Technically, we compare the mean and median of typea as well as generating a boxplot with the function multiBP() to get a rough threshold, which we use for the jitterplot multiJP to identify outliers. Moreover, we establish two controlling-models in the glm() environment. The first one consists of typea and the second one adds age. Comparing a logistic regression only containing typea as explanatory variables with a model where we control for age shows, that the p-value drastically decrease in the second case (see table 4 and 5).

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.8447	0.5607	-3.29	0.0010	**
typea	0.0226	0.0103	2.20	0.0275	*
Significance codes	0 ⇒ ***	0.001 ⇒ **	0.01 ⇒ *		
AIC	595.12				

Table 4: The summary statistics of the logistic regression of typea with the explanandum chd (see related Quantlet SAHeart_Q4_Logistic_Regression).

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-5.9170	0.8414	-7.03	0.0000	***
typea	0.0396	0.0115	3.44	0.0006	***
age	0.0702	0.0091	7.75	0.0000	***
Significance codes	0 ⇒ ***	0.001 ⇒ **	0.01 ⇒ *		
AIC	519.04				

Table 5: The summary statistics of the logistic regression of typea and age with the explanandum chd (see related Quantlet SAHeart_Q4_Logistic_Regression).

Thus, we can conclude that age serves as a cofounder here regarding typea and chd. This is sometimes called an *incidental cancellation*. Therefore, we will keep typea and select m2 as the best model here, since it has the lowest AIC value as well as the highest significant coefficients, which can be regarded at table 6.

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-6.4464	0.9209	-7.00	0.0000	***
tobacco	0.0804	0.0259	3.11	0.0019	**
age	0.0505	0.0102	4.94	0.0000	***
famhist	0.9082	0.2258	4.02	0.0001	***
ldl	0.1620	0.0550	2.95	0.0032	**
typea	0.0371	0.0122	3.05	0.0023	**
Significance codes	0 ⇒ ***	0.001 ⇒ **	0.01 ⇒ *		
AIC	487.69				

Table 6: The summary statistics of the logistic regression of m2 (see related Quantlet SAHeart_Q4_Logistic_Regression).

4.3 Leave-One-Out Cross-Validation

After calculating different possible models, we now want to check their out-of-sample performance as described in section 2.3 as another measure for the quality of the models and hence, for model selection.
For this purpose we wrote the function loo_cv, which takes two arguments. The first, model is an obligatory argument that needs to be a glm or lm object. These kind of objects are returned by the eponymous functions as introduced in section 4.2. The second, cutoff is an optional argument that marks the threshold whether an observation shall be allocated to class 0 or 1, since the prediction for an observation from a logistic regression model can be interpreted as a probability. The default is 0.5, i.e. if the probability to suffer from coronary heart disease is higher than 0.5 this observation will be allocated to class 1.
The underlying dataset of the input model and the formula of the regression model are read from the glm object. These objects are handled like a data.frame. Thus, we can easily access different information simply using the $-notation. Generally, this makes these objects to be easily used by other functions, e.g. from different packages, that perform tests, forecasting, interval calculations, graphical representations, etc. Furthermore, the number of rows is calculated and an empty vector is created using the vector command. Here, the mode, that is the type of the content, and the length are specified. The reason for this empty vector is that in the next step, we will write to it row by row. If the vector is already specified, we can use simple index notation for that. Of course, this is only possible in cases where we know the number of rows – or more generally entries – that will be written to the vector in advance. Another possibility is to create a vector or a list that will be appended on the fly in each iteration step. However, the first solution is recommendable because if the resulting vector is used in further operations its length/dimension might be important for functionality, e.g. if matrix multiplication were used. This could cause errors that might be difficult to track. Knowing the length/dimension allows to write better assessable upcoming code.
Afterwards, we are running over each observation fitting the model that was given into the loo_cv function using a for() loop. Note that the expression 1:N creates a

sequence $1, \ldots, N$ on the fly from which we take every element to iterate over, saved to the variable i in each step, so we can refer to it. In general, we could specify an arbitrary list instead of 1:N to iterate over each of its elements, e.g. the letters of the alphabet.

```
loo_cv <- function(model, cutoff = 0.5){

  cur.data <- model$data
  cur.formula <- model$formula
  N <- nrow(cur.data)

  prediction <- vector(mode = "numeric", length = N)

  for(i in 1:N){
    cur.model <- glm(cur.formula, family = binomial(link = "logit"),
                     data = cur.data[-i,])
    prediction[i] <- predict(cur.model, newdata = cur.data[i,], type = "response")
  }

  correct <- cur.data$chd == (prediction > cutoff)

  accuracy <- sum(correct)/N

  return(accuracy)
}


loo_cv(m0)
loo_cv(m2)
loo_cv(m3)
```

Code 15: Snippet of Quantlet SAHeart_Q5_Leave-One-Out_Cross-Validation

Again, we use the glm() function as described in section 4.2. The formula that was read from the input model is inserted here as well as the data. In each iteration step the observation i is removed from the dataset and the model fitted without it. Again, this is easily achieved using index notation. With -i a specific row – or column if it is used on the right side of the comma – is removed. Then, the probability of suffering from coronary heart disease for observation i is predicted using this model and written into the previously for this purpose specified vector. For the prediction of an observations using an already fitted model we use the predict() command that need the model as an glm object and a set of the respective explanatory variables for these observations as input. Furthermore the type argument needs to be set. response means in this case that we want to get the output in terms of a fitted value, i.e. as a probability in case of logistic regression. To be more precise, the input of a glm object indicates that the function predict.glm() will be called that can handle regression model in particular. The predict() function can handle several different object types that are suitable for predictions. Hence, there are several parameters that may be set or passed additionally to this function. But just giving a specific object type as input suffices here to get the desired output, what makes it easy to use in a lot of applications.

After the loop is done, the vector prediction contains the fitted values for each observation. Now, we check for each observation if the threshold given by cutoff is exceeded. If so, this yields a TRUE and otherwise a FALSE value. These values are in turn compared to the actual value of chd. Thus, we obtain the number of correct predictions by summing this up as well as the prediction accuracy by taking the proportion of all predictions. Returned is the prediction accuracy, in this case called the out-of-sample performance – since we are simulating to test the model on new data, as described in section 2.3.

model	out-of-sample performance
m0	.729
m2	.736
m3	.717

Table 7: The out-of-sample performance for three different models from leave-one-out cross-Validation (see related Quantlet SAHeart_Q5_Leave-One-Out_Cross-Validation).

The loo_cv function can now easily be used, simply inserting the models calculated in section 4.2. Table 7 shows the results. The model m2 has the best out-of-sample performance. This is in line with the results of section 4.2 that were driven by regression analysis using the Akaike information criterion and significances of the coefficients as well as theoretical aspects. However, we still may improve our model.

4.4 Cook's Distance

A further approach to improve the model is to find single observations that may influence the coefficients of the regression model in an inappropriate high manner. In section 3.2, we already saw some observations that are suspicious. E.g. the observation with the id 21 that is on the one hand very young, 17 years old, and has quite low values on all important risk factor, but on the other hand suffers from coronary heart disease, as can be seen from figure 6. This is very unlikely and the question arises if there happened a mistake during coding or if there are other reasons for that, e.g. some variables that are not represented in the dataset. Even if we say that this might be a case that occurs with very low probability in the population and therefore might be represented in a random sample, the sample at hand is not that large, so that such occurrences have a larger impact on the estimates of the regression coefficients than we would assume for the population. In short, such inappropriate high impacts level out in large samples but can have too much weight due to small sample sizes. In this case this might lead to underestimating the influence of the single risk factors.

An easy way that helps out here is to exclude the respective observations. But one needs to be cautious. The other way around there is the risk to exclude too many observations that don't really fit the needs of the models and thus over-fitting the model to the given data. Or, as it is sometimes put, to fit the data to the model instead of fitting the model to the data.

A straightforward method that helps to detect such observations is the Cook's Distance. We wrote a function that calculates the Cook's Distance for each observation and plots the results to directly spot influential observations at first glance. The function cookPlot takes three arguments. As in section 4.3, model needs to be a glm or lm object. The optional argument threshold can be used to set the threshold above which the Cook's Distance values are denoted as influential or at least as suspiciously high. And the third parameter PDFpath is again used to directly save the plot to the local computer.

```r
cookPlot <- function(model, threshold = NULL, PDFpath = NULL){

  cd <- cooks.distance(m2)

  if(!is.null(threshold)){
    additional_params <- list(geom_text(data = data.frame(id = SAheart$id,
                                                           cd = cd)[cd >= threshold,],
                                        aes(label = id), colour = 'firebrick',
                                        size = 3, vjust = -0.5),
                              geom_hline(yintercept = threshold, color = 'firebrick'))
  }else{
    additional_params <- NULL
  }

  final.plot <- ggplot(data = data.frame(id = SAheart$id, cd = cd),
                       mapping = aes(x = as.factor(id), y = cd)) +
    geom_bar(stat = "identity", width = 0.1, color = "black") +
    theme_classic() +
    labs(x = "", y = "Cook's D") +
    theme(axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
    scale_y_continuous(expand = c(0, 0), limits = c(0, max(cd)+0.002)) +
    additional_params

  plot(final.plot)

  if(!is.null(PDFpath)){
    ggsave(filename = "/CooksDplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }
```

Code 16: Snippet of Quantlet SAHeart_Q6_Cooks_Distance

For the calculation of the Cook's Distance values we use the function cooks.distance() from the stats-package that is loaded by default. Like the predict() command from section 4.3 this function takes a glm or lm object as input and calls the respective function cooks.distance.glm() or cooks.distance.lm(), respectively.[4] The output is a vector containing the Cook's Distance values for each observation that was given in the input model.

Furthermore, we check whether the threshold parameter has been specified using an if condition. A very nice feature of ggplot() is that we can append new layers and parameters by passing a list. Each element of the list will be handled as they were combined with +, what is the usual ggplot()-notation for adding layers and parameters. If this list is empty, i.e. NULL, it is just ignored. This makes it very easily possible, as it is done here, to add additional elements to the graphic on conditions. Since ggplot() is very flexible and comprehensive in parameters and graphic types there is a variety of possibilities to combine them with conditions. That makes it a remarkable tool for graphics. However, here we add two things to the graphic. First, a horizontal line marking the threshold with the geom_hline command and second the id of each observation that exceeds this threshold using the geom_text command. This can been seen in Figure 5. The data necessary for the labeling is here passed on the fly as a data.frame consisting of the id and the respective Cook's Distance values given that the value exceeds the threshold, using index notation. The vjust option, for *vertical adjustment*, adds some space between the bar an the label for a better appearance.

The commands to create the graphic have mostly been described above already. But notable is the scale_y_continuous command which is used to set the limits of the y-axis since we need some extra space at the top in case the labels are added. This is achieved by adding a little space to the maximum possible value, that is easily obtained with the function max(). And setting the expand parameter to c(0,0) removes the gap between the plot area and x-axis that is set by default. Also note how easy the list additional_params is added here.

[4]Note that in case of Gaussian regression models equation (10) can be expressed analytically and therefore calculated in a more efficient way. For the glm case an algorithm for approximation is used (for details see, Williams, 1987).

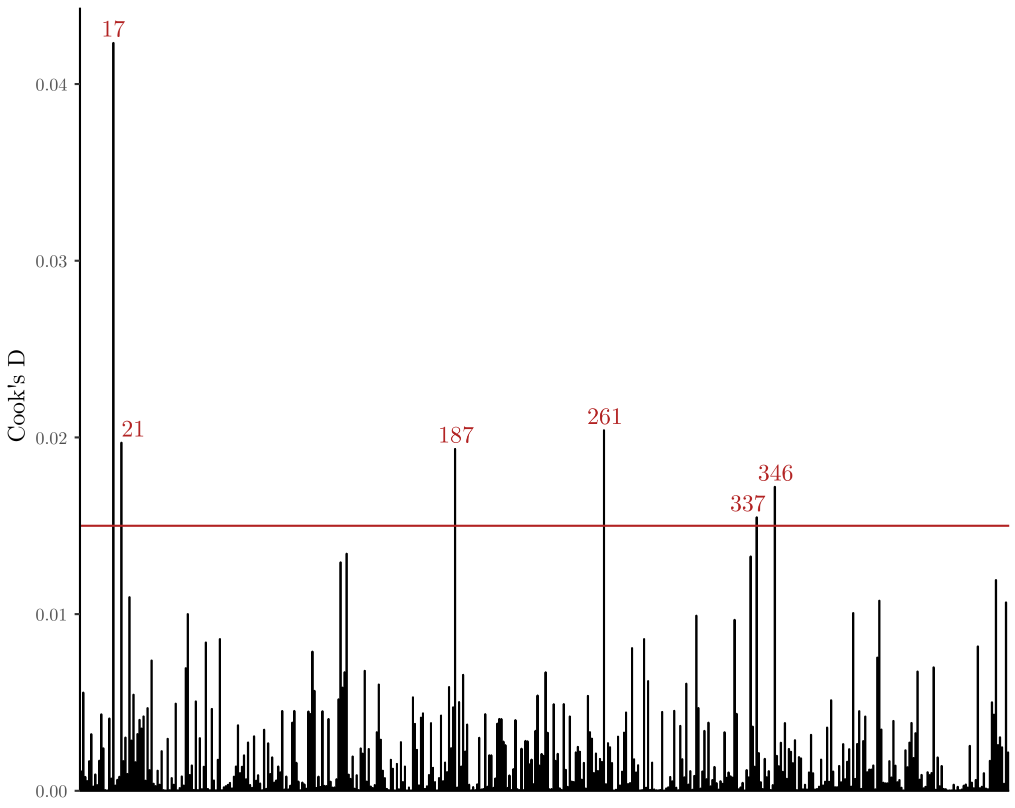

Figure 5: The Cook's Distance values for every observation. The red line marks the threshold above which the observations are denoted as influential. Those observations are labeled with their id (see related Quantlet SAHeart_Q6_Cooks_Distance).

The function doesn't return anything and just plots the graphic. To decide where the threshold should lie, we just ran cookPlot first without specifying the threshold parameter and obtained it as a visual estimate. Then, we run the function again with the threshold specified.

```
cookPlot(m2, threshold = 0.015, PDFpath = PDFpath)

cd <- cooks.distance(m2)

obslist <- SAheart$id[cd >= 0.015]
varlist <- c("typea", "tobacco", "age", "ldl", "famhist")
multiJP(varlist, obslist, label = TRUE, PDFpath = PDFpath)
```

Code 17: Snippet of Quantlet SAHeart_Q6_Cooks_Distance

We can see from figure 5 that there are six suspicious observations, including the observations 21 and 261 that we already spotted above. Especially observation 17 seems to have a high influence. To not just naively remove these observations on first movement, we use the multiJP function from above to get some more insights on these observations.

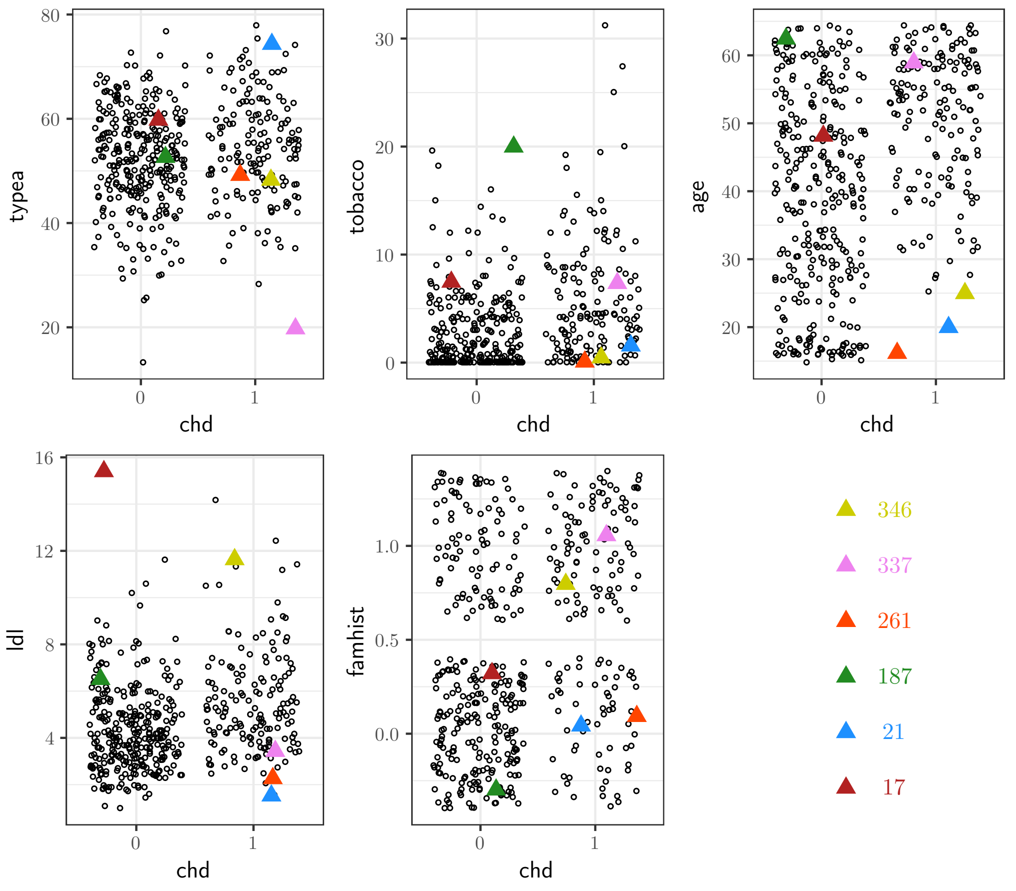

Figure 6: The distributions of the covariates over chd. Highlighted are the observations that have been detected as influential with Cook's Distance (see related Quantlet SAHeart_Q6_Cooks_Distance).

Figure 6 shows the behavior of these observation on the set of the explanatory variables we choose for our model. Observation 17 has high values – above the means – on typea, tobacco and age but no occurrence of coronary heart disease. This is not that conspicuous so far. But the value for ldl is very high compared to the other observations. As already mentioned above, we could ask if this is a mistake or just a rarely occurrence in the sample. However, for sake of not underestimating the single risk factors, what is obviously more fatal than overestimation – since in case of detection of diseases or risk factors for such, we rather take false positives than false negatives – we decided to exclude this specific observation. Also, for the same reasons, observations 21 and 261 are excluded. observation 261 shows very low values on all risk factors especially age. Observation 21 has also very low values on all these variables except for typea. Actually here, the value is pretty high. But this variable is very controversial as already mentioned in section 3.1. Therefore, we still decided to drop it.

```r
## Drop all very influencial obs for model calc
  obslist <- SAheart$id[cd >= 0.015]
  m2.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
              family = binomial(link = "logit"),
              data = SAheart[!(SAheart$id %in% obslist),])
  summary(m2.1)
  loo_cv(m2.1)

# Drop only the influencial points that are very suspiciuous in the graphics
# and whose excluding can be justified with theory
  obslist <- c("17", "261", "21")
  m2.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
              family = binomial(link = "logit"),
              data = SAheart[!(SAheart$id %in% obslist),])
  summary(m2.2)
  loo_cv(m2.2)
```

Code 18: Snippet of Quantlet SAHeart_Q6_Cooks_Distance

Observation 187 is the most controversial case here. On the one hand, are the values for tobacco and age very high, but not abnormally high compared to other observations. On the other hand, the other variables do not really suggest the occurrence of coronary heart disease. So, getting old and smoking alone while not suffering from coronary heart disease is probably not that unusual. This is why we did not remove this observation. Also in the case of the observations 337 and 346 we didn't found them conspicuously enough to drop them.
In the next section we will use these insights to refit the models without these observations and evaluate the regression results as well as the out-of-sample performance from the loo_cv function.

4.5 Results

Throughout this paper we developed the model on theoretical reflections and graphical exploration resulting in a logistic regression model with a plausible set of explanatory variables, namely the model m2. Lastly, we spotted some observations that may be excluded from the dataset to refit the model without them. Thus, hopefully obtaining even better estimates. This is what we will do in this last section to select the final model that we would suggest from this dataset.
Table 8 shows the regression outputs for three models as well as the out-of-sample performance obtained from the loo_cv function. First, the model m2 is presented here again for comparison reason. Model m2.1 uses the same explanatory variables

	m2		m2.1		m2.2	
	coeff.	std. err.	coeff.	std. err.	coeff.	std. err.
Exclude observations	None		17, 21, 187, 261, 337, 346		17, 21, 261	
Coefficients						
constant	-6.446^{***}	0.921	-7.241^{***}	0.992	-6.853^{***}	0.959
tobacco	0.080^{**}	0.026	0.090^{***}	0.027	0.088^{**}	0.027
age	0.051^{***}	0.010	0.057^{***}	0.011	0.053^{***}	0.011
famhist	0.908^{***}	0.226	0.873^{***}	0.232	0.923^{***}	0.230
ldl	0.162^{**}	0.055	0.196^{***}	0.060	0.205^{***}	0.059
typea	0.037^{**}	0.012	0.042^{***}	0.013	0.037^{**}	0.012
AIC	487.690		464.500		472.150	
out-of-sample performance	0.736		0.745		0.747	

$^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$

Table 8: Results of the Logistic Regression models fitted on different sets of observations. All models use the same set of explanatory variables as chosen for m2 in section 4.2 (see related Quantlet SAHeart_Q4_Logistic_Regression, Quantlet SAHeart_Q5_Leave-One-Out_Cross-Validation and Quantlet SAHeart_Q6_Cooks_Distance) .

but excluding all observations that exceeded the threshold in Figure 5. And m2.2 also uses the same explanatory variables but here all observations are excluded that could be excluded for theoretical reasons as discussed in section 4.4. The models m2.1 and m2.2 perform better on all indicators that we used for model selection here than model m2. The significance levels are better as well as especially the Akaike information criterion and the out-of-sample performance.

At first glance, one would select m2.1 as the best model. But let us first reflect on what these measures mean and what they are based on. The significance level is here a measure of the certainty with which we can reject the null hypotheses that the true coefficient is zero. Of course, the higher this certainty the better. But the benefit gets smaller if we have already quite good p-values for all coefficients as it is the case here, actually at least $p < 0.01$. So, attaining even higher p-values from this start point is obviously not a sufficient logic here to select a model. Especially, and that is the most important point here, when this is attained by dropping observations that does not very well fit to the model. Because going on that way means from the perspective of equation (6) to successively remove the ϵ_i that are highest and thus, artificially shrinking the estimate of the variance $\hat{\sigma}^2_\epsilon$. Consequently, since the ϵ_i are the only stochastic components of the regression model, this affects the standard errors of the estimates and thus the p-values. And this is literally fitting the data to the model.

The same applies to the Akaike information criterion since it refers to the likelihood function of the model. And, for another reason, also to the out-of-sample performance since the loo_cv function only uses the observations that are passed through the input model. Therefore, the scope is still restricted to itself and that will automatically increase the prediction accuracy. However, to cut a long story short, the measures for model selection are of no value if theoretical aspects are ignored. Thus, unless model m2.1 looks appealing, we choose m2.2 as our final model since we can justify this theoretically as has been done in section 4.4.

To compare this to the Fisher's linear discriminant, where we achieved a prediction accuracy of 0.695, we managed to improve the prediction accuracy to 0.750, or by means of the out-of-sample performance, to 0.747. That is an increase of ca. 0.05. In addition, the logistic regression model provides an overview of how strong the single variables effect the occurence of coronary heart disease. This is a great advantage of the Fisher's linear discriminant method especially if the model is not just viewed from the perspective of classification but rather from the view of the risk factors itself.

5 Conclusion

As we already summarized the statistical results in the chapter before, the following will concentrate on our work with *R* and the real-world implications of the statistical results.

In the course of this paper we used different features of *R*. We loaded extensions and implemented code that deals with errors and shows easily understandable warning messages to the user. We analyzed data with the statistical tools of *R* and visualized the results in numerous complex graphics. Some of the code was written in form of functions, so that is usable for repeated cases by redefining the parameters. We also calculated the Fisher's linear discriminant from the scratch to show some fundamental functions of *R*, e.g. working with matrices.

Content-wise, our goal was to find the major risk factors of heart disease. Our descriptive analysis revealed, that there is a noteworthy correlation of age, sbp, tobacco, adiposity and ldl levels with the occurrence of heart disease. The regression analysis gave further details, which factors how strongly influence the occurrence of heart disease. Using the leave-one-out cross-validation and Cook's distance, we determine model m2.2 as our best model as it has the highest prediction accuracy. Its real-world implications are that behavioral factors like smoking and aggressive behavior play an important role, but there are also genetic factors like the family history (famhist) of heart disease. Additionally, cholesterol levels (ldl) and age are major predictors for the occurrence of heart disease. It is always dangerous to suggest concrete measures since the correlation doesn't necessarily implicates a causal relationship. It nevertheless seems to be a safe proposition that health organizations should intensify campaigns against smoking and should recommend mandatory health checks for people at a certain age. These results are in line with the findings of Rossouw et al. (1983). We were not able to compare the detailed results as the data set in *R* only is a non-representative subset of the data Rossouw et al. (1983) used. In terms of conclusions, they additionally found that there should be health checks for people with obesity. As most participants exhibited at least one major risk factor, they even proposed to implement preventive measures for the whole community, which may be unrealistically costly for everything that goes further than information. Current research strongly suggests focusing on individuals with a high risk-score with the major factors tobacco, alcohol and overweight (obesity and adiposity) (cf. Putadechakum et al. (2014)).

References

Cook, R. D. (1977). Detection of influential observations in linear regression. *Technometrics*, 19(1):15–18.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification.* Wiley & Sons, New York.

Putadechakum, S., Leelahakul, V., Rodjinda, P., Phanachet, P., and Roongpisuthipong, C. (2014). Cardiovascular risk factors and 10-year risk for coronary heart disease in thai adults. www.ijsrp.org/research-paper-0514/ijsrp-p29119.pdf . Retrieved on March 24th, 2018.

R-Foundation (2018). What is R? https://www.r-project.org/about.html . Retrieved on March 24th, 2018.

Rossouw, J. E., Plessis, J. P. D., Benad, A. J. S., Jordaan, P. C. J., Kotz, J. P., Jooste, P. L., and Ferreira, J. J. (1983). Coronary risk factor screening in three rural communities – the coris baseline study. *South African Medical Journal*, 64.

Spokoiny, V. and Dickhaus, T. (2015). *Basics of modern mathematical statistics.* Springer, Berlin/Heidelberg.

Wasserman, L. (2013). *All of statistics: a concise course in statistical inference.* Springer Science & Business Media, New York.

Wickham, H., Francois, R., Henry, L., Müller, K., and Wickham, M. H. (2017). Package "dplyr". ftp://cran.de.r-project.org/pub/R/web/packages/dplyr/dplyr.pdf . Retrieved on March 24th, 2018.

Williams, D. A. (1987). Generalized linear model diagnostics using the deviance and single case deletions. *Applied Statistics*, 36:181–191.

World Health Organization (2017). The top 10 causes of death - fact sheets. http://www.who.int/mediacentre/factsheets/fs310/en/. Retrieved on March 24th, 2018.

A Tables

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-6.1507	1.3083	-4.70	0.0000	***
sbp	0.0065	0.0057	1.14	0.2564	
tobacco	0.0794	0.0266	2.98	0.0028	**
ldl	0.1739	0.0597	2.92	0.0036	**
adiposity	0.0186	0.0293	0.63	0.5257	
famhist	0.9254	0.2279	4.06	0.0000	***
typea	0.0396	0.0123	3.21	0.0013	**
obesity	-0.0629	0.0442	-1.42	0.1551	
alcohol	0.0001	0.0045	0.03	0.9784	
age	0.0452	0.0121	3.73	0.0002	***
Significance codes	0 $\Rightarrow$ ***	0.001 $\Rightarrow$ **	0.01 $\Rightarrow$ *		
AIC	492.14				

Table 9: The summary statistics of the logistic regression of m0 (see related Quantlet SAHeart_Q4_Logistic_Regression).

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-6.3603	0.9356	-6.80	0.0000	***
tobacco	0.0804	0.0259	3.10	0.0019	**
age	0.0529	0.0113	4.67	0.0000	***
famhist	0.9097	0.2259	4.03	0.0001	***
ldl	0.1724	0.0592	2.91	0.0036	**
typea	0.0371	0.0122	3.05	0.0023	**
adiposity	-0.0094	0.0192	-0.49	0.6229	
Significance codes	0 $\Rightarrow$ ***	0.001 $\Rightarrow$ **	0.01 $\Rightarrow$ *		
AIC	489.44				

Table 10: The summary statistics of the logistic regression of m1.1 (see related Quantlet SAHeart_Q4_Logistic_Regression).

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-5.7027	1.0764	-5.30	0.0000	***
tobacco	0.0800	0.0260	3.08	0.0021	**
age	0.0521	0.0102	5.09	0.0000	***
famhist	0.9161	0.2264	4.05	0.0001	***
ldl	0.1837	0.0582	3.16	0.0016	**
typea	0.0383	0.0122	3.13	0.0017	**
obesity	-0.0376	0.0291	-1.29	0.1964	
Significance codes	0 ⇒ ***	0.001 ⇒ **	0.01 ⇒ *		
AIC	487.98				

Table 11: The summary statistics of the logistic regression of m1.2 (see related Quantlet SAHeart_Q4_Logistic_Regression).

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-4.2043	0.4983	-8.44	0.0000	***
tobacco	0.0807	0.0255	3.16	0.0016	**
age	0.0440	0.0097	4.52	0.0000	***
famhist	0.9241	0.2232	4.14	0.0000	***
ldl	0.1676	0.0542	3.09	0.0020	**
Significance codes	0 ⇒ ***	0.001 ⇒ **	0.01 ⇒ *		
AIC	495.44				

Table 12: The summary statistics of the logistic regression of m3 (see related Quantlet SAHeart_Q4_Logistic_Regression).

B Code

B.1 Descriptive Statistics

B.1.1 Quantlet 1 - Data Preparation

```
## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")
```

```

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}


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


## Variables of dataset
# sbp          systolic blood pressure
#
# tobacco      cumulative tobacco (kg)
#
# ldl          low density lipoprotein cholesterol
#
# adiposity    a numeric vector
#
# famhist      family history of heart disease, a factor with levels Absent Present
#
# typea        type-A behavior
#
# obesity      a numeric vector
#
# alcohol      current alcohol consumption
#
# age          age at onset
#
# chd          response, coronary heart disease


##################################
##                              ##
##     Description of dataset   ##
##                              ##
##################################


## Some overview stuff

glimpse(SAheart)
str(SAheart)

head(SAheart, 5)

summary(SAheart)


## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())


## famhist
head(SAheart$famhist, 10)
head(as.numeric(SAheart$famhist), 10)
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1
```

Code 19: Quantlet SAHeart_Q1_Data_Preparation

B.1.2 Quantlet 2 - Descriptive Statistics

```r

## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}



## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())


## famhist
head(SAheart$famhist, 10)
head(as.numeric(SAheart$famhist), 10)
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1



##################################
##                              ##
##     Exploratory Analysis     ##
##                              ##
##################################


## Correlation matrix (using package corrplot)
pdf(file = paste0(PDFpath,"/Corrplot.pdf"), width = 7, height = 8)
corr_matrix <- cor(select(SAheart, chd, everything(), -id))
colset <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot.mixed(corr_matrix,
               upper = "ellipse",
               lower = "number",
               number.cex = .7,
               upper.col = colset(10),
               lower.col = "black",
               tl.col = "black",
               tl.pos = "lt")
dev.off()


## Function to show Boxplots of covariates over chd
multiBP <- function(varlist, PDFpath = NULL){
  plot.list <- lapply(varlist, function(ivar){
    ggplot(data = SAheart, mapping = aes(x = as.factor(chd), y = get(ivar))) +
      geom_boxplot(stat = "boxplot", position = "dodge") +
      theme_bw() +
      xlab("chd") +
      ylab(ivar)
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3
  }

  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)

  if(!is.null(PDFpath)){
    ggsave(filename = "/BPplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }
}


varlist <- c("sbp", "tobacco", "age", "ldl", "adiposity")
multiBP(varlist, PDFpath)


## Suspicious outlier
SAheart$id[SAheart$age <= 20 & SAheart$chd == 1]

subset(SAheart, select = varlist, id == 261)
subset(SAheart, select = varlist, id == 21)
summary(subset(SAheart, select = varlist, chd == 1))

## Function to visualize behaviour of single observations
multiJP <- function(varlist, obslist = 0, PDFpath = NULL, label = FALSE){


  if(label){
    varlist <- c(varlist, "legend")
  }

  color.list <- c("firebrick", "dodgerblue", "forestgreen",
                  "orangered", "violet", "yellow3")

  plot.list <- lapply(varlist, function(ivar){
    if(ivar == "legend"){
      hackdata <- data.frame(id = obslist,
                             position = seq(1,length(obslist)),
                             xvalue = 0.11)
      legend <- ggplot(data = hackdata, mapping = aes(x = xvalue, y = position)) +
        geom_point(colour = color.list[1:length(obslist)], shape = 17, size = 3) +
        theme_classic() +
        theme(axis.text = element_blank(), axis.ticks = element_blank(),
              axis.line = element_blank()) +
        labs(x = "", y = "") +
        scale_y_continuous(expand = c(0, 0), limits = c(0, max(hackdata$position)+1)) +
        scale_x_continuous(limits = c(0.1, 1)) +
        geom_text(data = hackdata, aes(label = id),
                  colour = color.list[1:length(obslist)], size = 3, hjust = -1)

    }else{

      ggplot(data = SAheart[!(SAheart$id %in% obslist),],
             mapping = aes(x = as.factor(chd), y = get(ivar))) +
        geom_jitter(shape = 1) +
        theme_bw() +
        xlab("chd") +
        ylab(ivar) +
        geom_jitter(data = SAheart[SAheart$id %in% obslist,],
                    colour = color.list[1:length(obslist)], shape = 17, size = 3) # +
    }
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3 }

  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)

  if(!is.null(PDFpath)){
    ggsave(filename = "/Jitterplot_outliers.pdf", path = PDFpath, device = "pdf", plot = final.plot)}}


varlist <- c("sbp", "tobacco", "age", "ldl", "adiposity")
multiJP(varlist, obslist = c(261))


## Also check for the other very young person:
multiJP(varlist, obslist = c(261, 21), label = TRUE, PDFpath = PDFpath)


## Check in general for obs that have chd but have very low values on all important variables
obslist <- (SAheart %>%
              filter(chd == 1,
                     sbp <= quantile(sbp, .25),
                     tobacco <= quantile(tobacco, .25),
                     age <= quantile(age, .25),
                     ldl <= quantile(ldl, .25),
                     adiposity <= quantile(adiposity, .25),
                     famhist == 0) %>%
              select(id))$id
```

Code 20: Quantlet SAHeart_Q2_Descriptive_Statistics

B.2 Inferential Statistics

B.2.1 Quantlet 3 - Fisher's Linear Discriminant

```r

## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}

## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1


##################################
##                              ##
##   Fisher Linear Discirminant ##
##                              ##
##################################


## Function for Fisher Linear Discriminant
fisher <- function(X, c, PDFpath = NULL){

  X <- as.matrix(X)

  X0 <- X[c == 0,]
  X1 <- X[c == 1,]

  n0 <- nrow(X0)
  n1 <- nrow(X1)

  m0 <- as.vector(colMeans(X0))
  m1 <- as.vector(colMeans(X1))

  M0 <- as.vector(rep(1, n0)) %*% t(m0)
  M1 <- as.vector(rep(1, n1)) %*% t(m1)

  X0_c <- X0 - M0
  X1_c <- X1 - M1

  S0 <- t(X0_c) %*% X0_c
  S1 <- t(X1_c) %*% X1_c

  S_w <- S0 + S1

  S_w_inv <- solve(S_w)

  w <- S_w_inv %*% (m0 - m1)

  m <- as.vector(colMeans(X))
  M <- as.vector(rep(1, nrow(X))) %*% t(m)
  X_c <- X - M

  z <- t(w) %*% t(X_c) < 0

  accuracy <- sum(t(z) == c) / length(z)

  Y0 <- X0 %*% w
  Y1 <- X1 %*% w

  Y <- rbind(cbind(Y0, rep(0,nrow(Y0))),
             cbind(Y1, rep(1,nrow(Y1))))
  Y <- as.data.frame(Y)
  colnames(Y) <- c("Y","c")

  final.plot <- ggplot(Y, aes(Y, fill = as.factor(c))) + geom_density(alpha = 0.2) +
    theme_classic() +
    labs(x = "", y = "") +
    theme(axis.text.x = element_blank(), axis.ticks.x = element_blank(),
          axis.text.y = element_blank(), axis.ticks.y = element_blank(),
          axis.line.y = element_blank()) +
      scale_fill_manual(values = c("firebrick", "dodgerblue"),
                        name = "chd", labels = c("no", "yes"))

  plot(final.plot)

  if(!is.null(PDFpath)){
    ggsave(filename = "/Densitiesplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }

  return(accuracy)
}


X <- select(SAheart, -id, -chd)
c <- SAheart$chd

fisher(X=X, c=c, PDFpath)
```

Code 21: Quantlet SAHeart_Q3_Fisher_Linear_Discriminant

B.2.2 Quantlet 4 - Logistic Regression

```

## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}



## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1

## Function to show Boxplots of covariates over chd
multiBP <- function(varlist, PDFpath = NULL){
  plot.list <- lapply(varlist, function(ivar){
    ggplot(data = SAheart, mapping = aes(x = as.factor(chd), y = get(ivar))) +
      geom_boxplot(stat = "boxplot", position = "dodge") +
      theme_bw() +
      xlab("chd") +
      ylab(ivar)
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3
  }

  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)
}

## Function to visualize behaviour of single observations
multiJP <- function(varlist, obslist = 0, PDFpath = NULL, label = FALSE){

  if(label){
    labellist <- obslist
    varlist <- c(varlist, "legend")
  }else{
    labellist <- ""
  }

  color.list <- c("firebrick", "dodgerblue", "forestgreen",
                  "orangered", "violet", "yellow3")

  plot.list <- lapply(varlist, function(ivar){
    if(ivar == "legend"){
      hackdata <- data.frame(id = obslist,
                             position = seq(1,length(obslist)),
                             xvalue = 0.11)
      legend <- ggplot(data = hackdata, mapping = aes(x = xvalue, y = position)) +
        geom_point(colour = color.list[1:length(obslist)], shape = 17, size = 3) +
        theme_classic() +
        theme(axis.text = element_blank(), axis.ticks = element_blank(),
              axis.line = element_blank()) +
        labs(x = "", y = "") +
        scale_y_continuous(expand = c(0, 0), limits = c(0, max(hackdata$position)+1)) +
        scale_x_continuous(limits = c(0.1, 1)) +
        geom_text(data = hackdata, aes(label = id),
                  colour = color.list[1:length(obslist)], size = 3, hjust = -1)

    }else{

      ggplot(data = SAheart[!(SAheart$id %in% obslist),],
             mapping = aes(x = as.factor(chd), y = get(ivar))) +
        geom_jitter(shape = 1) +
        theme_bw() +
        xlab("chd") +
        ylab(ivar) +
        geom_jitter(data = SAheart[SAheart$id %in% obslist,],
                    colour = color.list[1:length(obslist)], shape = 17, size = 3) # +
    }
  })

  if(length(plot.list) <= 3){
    ncol <- length(plot.list)
  }else{
    ncol <- 3
  }
  final.plot <- grid.arrange(grobs = plot.list, ncol=ncol)
}

##################################
##                              ##
##      Logistic Regression     ##
##                              ##
##################################


## M0: (naive) Full model
m0 <- glm(chd ~ ., family = binomial(link = "logit"),
          data = select(SAheart, -id))
summary(m0)


## M1: Wihtout sbp, alcohol; and exclude either adiposity or obesity
m1.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea + adiposity,
            family = binomial(link = "logit"), data = SAheart)
summary(m1.1)

m1.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea + obesity,
            family = binomial(link = "logit"), data = SAheart)
summary(m1.2)



## M2: Exclude sbp, alcohol, adiposity, obesity
m2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
          family = binomial(link = "logit"), data = SAheart)
summary(m2)


## M3: For sake of simplicity of models we try to exclude
## typea, since effect seems to be still very low
m3 <- glm(chd ~ tobacco + age + famhist + ldl,
          family = binomial(link = "logit"), data = SAheart)
summary(m3)
```

```r



## typea: mediator effect via age

mean(SAheart$chd[SAheart$typea >= median(SAheart$typea)])
mean(SAheart$chd[SAheart$typea < median(SAheart$typea)])

multiBP("typea")

obslist <- (SAheart %>%
               filter(chd == 1 & typea < 30) %>%
               select(id))$id

varlist <- c("sbp", "tobacco", "age", "ldl", "typea", "famhist")
multiJP(varlist, obslist, label = TRUE)


summary(glm(chd ~ typea, family = binomial(link = "logit"), data = SAheart))
summary(glm(chd ~ typea + age, family = binomial(link = "logit"), data = SAheart))
```

Code 22: Quantlet SAHeart_Q4_Logistic_Regression

B.2.3 Quantlet 5 - Leave-One-Out Cross-Validation

```r

## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}


## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1


## M0: (naive) Full model
m0 <- glm(chd ~ ., family = binomial(link = "logit"),
          data = select(SAheart, -tobacco, -id))

## M1: Wihtout sbp, alcohol; and exclude either adiposity or obesity
m1.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea + adiposity,
            family = binomial(link = "logit"), data = SAheart)

m1.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea + obesity,
            family = binomial(link = "logit"), data = SAheart)

## M2: Exclude sbp, alcohol, adiposity, obesity
m2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
          family = binomial(link = "logit"), data = SAheart)

## M3: For sake of simplicity of models we try to exclude typea, since effect seems to be still very low
m3 <- glm(chd ~ tobacco + age + famhist + ldl,
          family = binomial(link = "logit"), data = SAheart)

```

```

##################################
##                              ##
##         loo-cv function      ##
##                              ##
##################################


loo_cv <- function(model, cutoff = 0.5){

  cur.data <- model$data
  cur.formula <- model$formula
  N <- nrow(cur.data)

  prediction <- vector(mode = "numeric", length = N)

  for(i in 1:N){
    cur.model <- glm(cur.formula, family = binomial(link = "logit"),
                     data = cur.data[-i,])
    prediction[i] <- predict(cur.model, newdata = cur.data[i,], type = "response")
  }

  correct <- cur.data$chd == (prediction > cutoff)

  accuracy <- sum(correct)/N

  return(accuracy)
}


loo_cv(m0)
loo_cv(m2)
loo_cv(m3)
```

Code 23: Quantlet SAHeart_Q5_Leave-One-Out_Cross-Validation

B.2.4 Quantlet 6 - Cook's Distance

```

## Check if required packages are installed, if not install and load
loadPKG <- function(pkg){
  if(!(pkg %in% installed.packages())){
    message(paste0("The required package \'",
                   pkg, "\'is currently not installed. ",
                   "It will now be installed to the default library path."))
    install.packages(pkg)
  }
  library(pkg, character.only = TRUE)
}

loadPKG("dplyr")
loadPKG("ggplot2")
loadPKG("gridExtra")
loadPKG("ElemStatLearn")
loadPKG("corrplot")
loadPKG("xtable")

data(SAheart)


## Specify Path to save graphic outputs
PDFpath <- "~/Desktop"


## check if PDFpath exists, if not choose current working directory
if(!dir.exists(PDFpath)){
  PDFpath <- getwd()
  message(paste0("The \'PDFpath\' that was specified does not exist. ",
                 "Instead, the graphic outputs will be saved to the current ",
                 "working directory. The current working directory is: \'",
                 getwd(),"\'"))
}



## Create id variable to easier handle single observations (using 'dplyr')
SAheart <- SAheart %>%
  mutate(id = row_number())
SAheart$famhist <- as.numeric(SAheart$famhist)
SAheart$famhist[SAheart$famhist == 1] <- 0
SAheart$famhist[SAheart$famhist == 2] <- 1



## M0: (naive) Full model
m0 <- glm(chd ~ ., family = binomial(link = "logit"),
          data = select(SAheart, -tobacco, -id))
```

```r


## M1: Wihtout sbp, alcohol; and exclude either adiposity or obesity
m1.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea + adiposity,
            family = binomial(link = "logit"), data = SAheart)


m1.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea + obesity,
            family = binomial(link = "logit"), data = SAheart)


## M2: Exclude sbp, alcohol, adiposity, obesity
m2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
          family = binomial(link = "logit"), data = SAheart)


## M3: For sake of simplicity of models we try to exclude typea, since effect seems to be still very low
m3 <- glm(chd ~ tobacco + age + famhist + ldl,
          family = binomial(link = "logit"), data = SAheart)

loo_cv <- function(model, cutoff = 0.5){

  cur.data <- model$data
  cur.formula <- model$formula
  N <- nrow(cur.data)

  prediction <- vector(mode = "numeric", length = N)

  for(i in 1:N){
    cur.model <- glm(cur.formula, family = binomial(link = "logit"),
                     data = cur.data[-i,])
    prediction[i] <- predict(cur.model, newdata = cur.data[i,], type = "response")
  }

  correct <- cur.data$chd == (prediction > cutoff)

  accuracy <- sum(correct)/N

  return(accuracy)
}

##################################
##                              ##
##        Cook's Distance       ##
##                              ##
##################################


cookPlot <- function(model, threshold = NULL, PDFpath = NULL){

  cd <- cooks.distance(m2)

  if(!is.null(threshold)){
    additional_params <- list(geom_text(data = data.frame(id = SAheart$id,
                                                          cd = cd)[cd >= threshold,],
                                        aes(label = id), colour = 'firebrick',
                                        size = 3, vjust = -0.5),
                              geom_hline(yintercept = threshold, color = 'firebrick'))
  }else{
    additional_params <- NULL
  }

  final.plot <- ggplot(data = data.frame(id = SAheart$id, cd = cd),
                       mapping = aes(x = as.factor(id), y = cd)) +
    geom_bar(stat = "identity", width = 0.1, color = "black") +
    theme_classic() +
    labs(x = "", y = "Cook's D") +
    theme(axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
    scale_y_continuous(expand = c(0, 0), limits = c(0, max(cd)+0.002)) +
    additional_params

  plot(final.plot)

  if(!is.null(PDFpath)){
    ggsave(filename = "/CooksDplot.pdf", path = PDFpath, device = "pdf", plot = final.plot)
  }

}


cookPlot(m2, threshold = 0.015, PDFpath = PDFpath)

cd <- cooks.distance(m2)

obslist <- SAheart$id[cd >= 0.015]
varlist <- c("typea", "tobacco", "age", "ldl", "famhist")
multiJP(varlist, obslist, label = TRUE, PDFpath = PDFpath)

## Drop all very influencial obs for model calc
  obslist <- SAheart$id[cd >= 0.015]
  m2.1 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
              family = binomial(link = "logit"),
              data = SAheart[!(SAheart$id %in% obslist),])
  summary(m2.1)
  loo_cv(m2.1)
```

```

# Drop only the influencial points that are very suspiciuous in the graphics
# and whose excluding can be justified with theory
  obslist <- c("17", "261", "21")
  m2.2 <- glm(chd ~ tobacco + age + famhist + ldl + typea,
              family = binomial(link = "logit"),
              data = SAheart[!(SAheart$id %in% obslist),])
  summary(m2.2)
  loo_cv(m2.2)
```

Code 24: Quantlet SAHeart_Q6_Cooks_Distance

C Declaration Of Authorship

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, March 30th 2018

Lukas Bargel, Adrian Rolf and Felix Vala

[illegible] [illegible] [illegible]