Exercises and Solutions - Analysis of High-Dimensional Data

Nicolas Städler

2024-05-08

Contents

1 Prerequisites

The data sets used in the following exercises can be downloaded from here. We recommend to install the following R packages: `tidyverse`, `knitr`, `caret`, `glmnet`, `MASS`, `lars`, `gbm`, `splines`, `randomForest`, `rpart`, `rpart.plot`, `ggpubr`, `survival`, `survminer`.

Let's load `knitr` and `tidyverse`:

```
library(tidyverse)
library(knitr)
```

2 Diabetes data and linear regression

The data that we consider consist of 442 diabetes patients, with the response of interest being a quantitative measure of disease progression one year after baseline. There are ten baseline variables — age, sex, body-mass index, average blood pressure, and six blood serum measurements — plus quadratic terms, giving a total of $p = 64$ features.

1. Read the diabetes data set `diabetes.rds` and make a histogram for the response variable `y`. Describe the distribution of the variable.
2. Create a scatterplot matrix for the 5 first variables in the data set. Use `pairs` or `ggpairs`.
3. Randomly assign patients to training and test data (use `sample`).
4. Run a univariate regression model with `bmi` as covariate. Study the `summary` output.
    - How do you interpret the regression coefficients for `bmi`?
    - What is the meaning of the *multiple R-squared*?
    - What is the *residual standard error*?
    - Generate a scatter plot of `y` against `bmi` and add the regression line with confidence band (use `geom_smooth`, `method="lm"`).
    - Draw the Tukey Anscombe plot and the QQ plot (check `?plot.lm`). What are these two plots telling us?
5. Run a multiple regression model using all covariates. Study the `summary` output.
    - What does change in the interpretation of the coefficient for `bmi`?
    - What do you conclude from the *multiple R-squared*?
    - Create a Tukey Anscombe plot and a QQ plot.
6. Calculate the RSS for both models. Write down your observation.
7. Compare the two models using the `anova` function. What do you conclude?

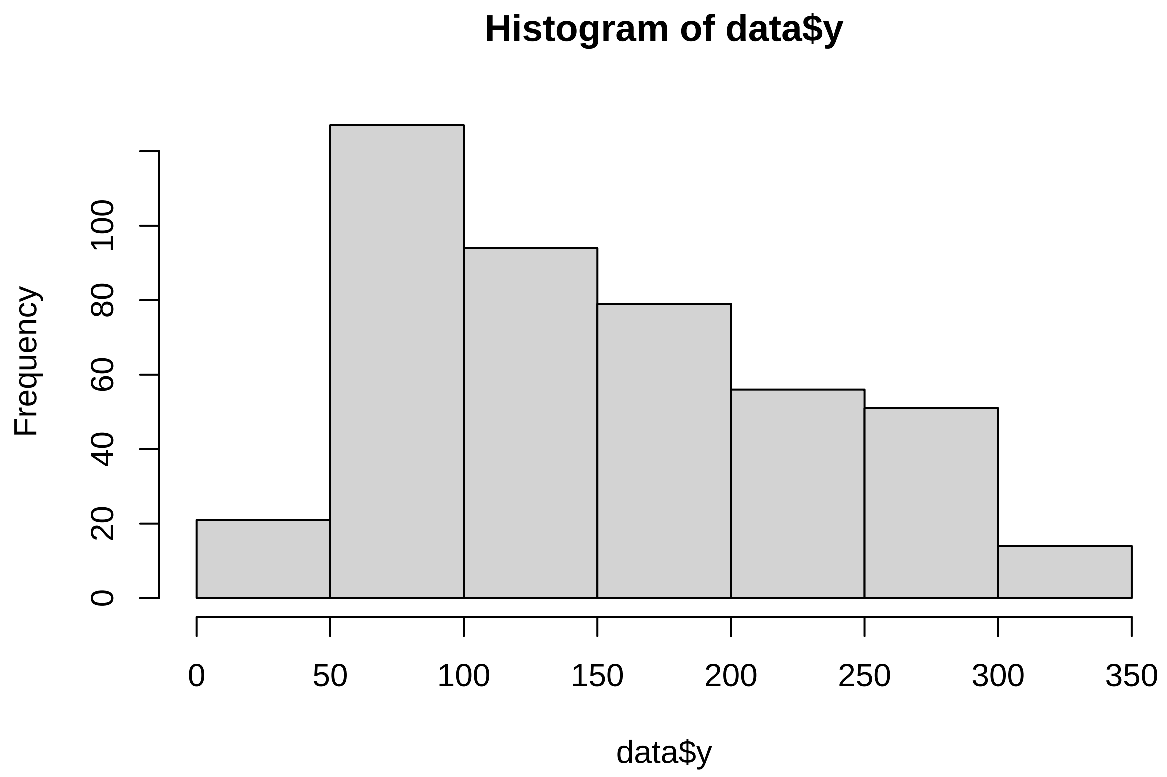
Solution to the exercise.

Read the data set.

```
diabetes <- readRDS(file="data/diabetes.rds")
data <- as.data.frame(cbind(y=diabetes$y,diabetes$x2))
colnames(data) <- gsub(":",".",colnames(data))
```

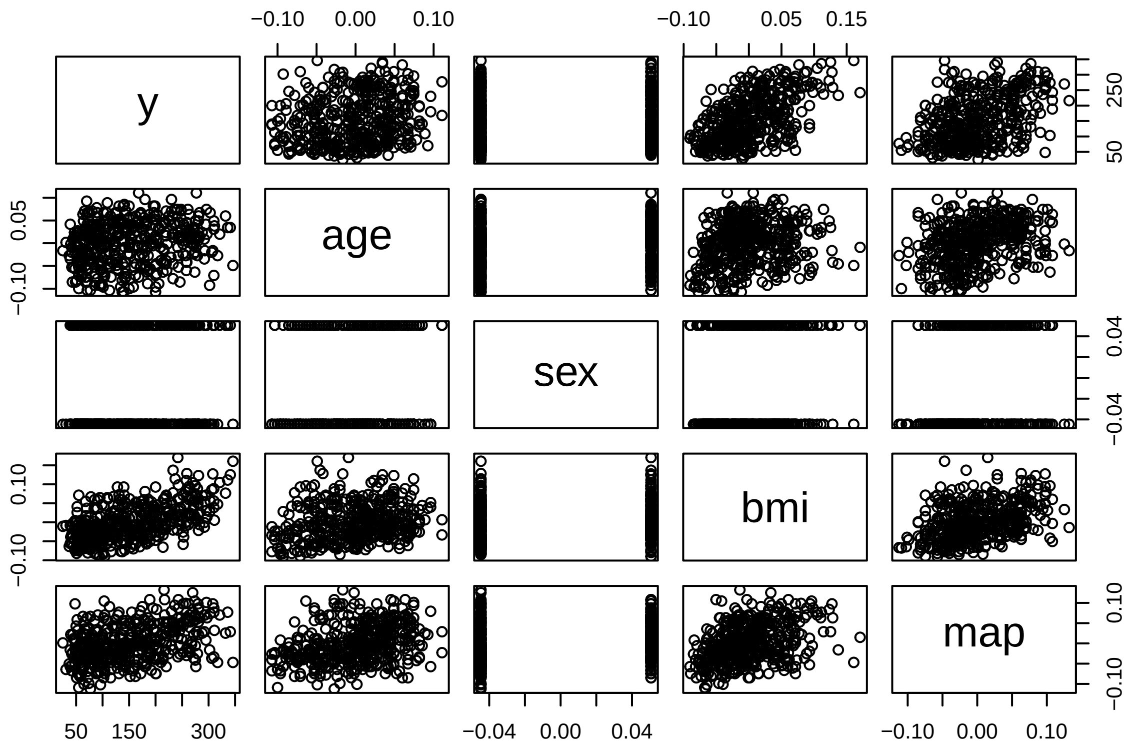
Generate a histogram of `y`.

```r
hist(data$y)
```



The distribution is right-skewed. Scatterplot matrix of the diabetes data.

```r
pairs(data[,1:5])
```



Create training and test data.

```
set.seed(1111)
train_ind <- sample(seq(nrow(data)),size=nrow(data)/2)
data_train <- data[train_ind,]
xtrain <- as.matrix(data_train[,-1])
ytrain <- data_train[,1]
data_test <- data[-train_ind,]
xtest <- as.matrix(data_test[,-1])
ytest <- data_test[,1]
```

Fit a univariate regression model.

```
fit1 <- lm(y~bmi,data=data_train)
summary(fit1)
```

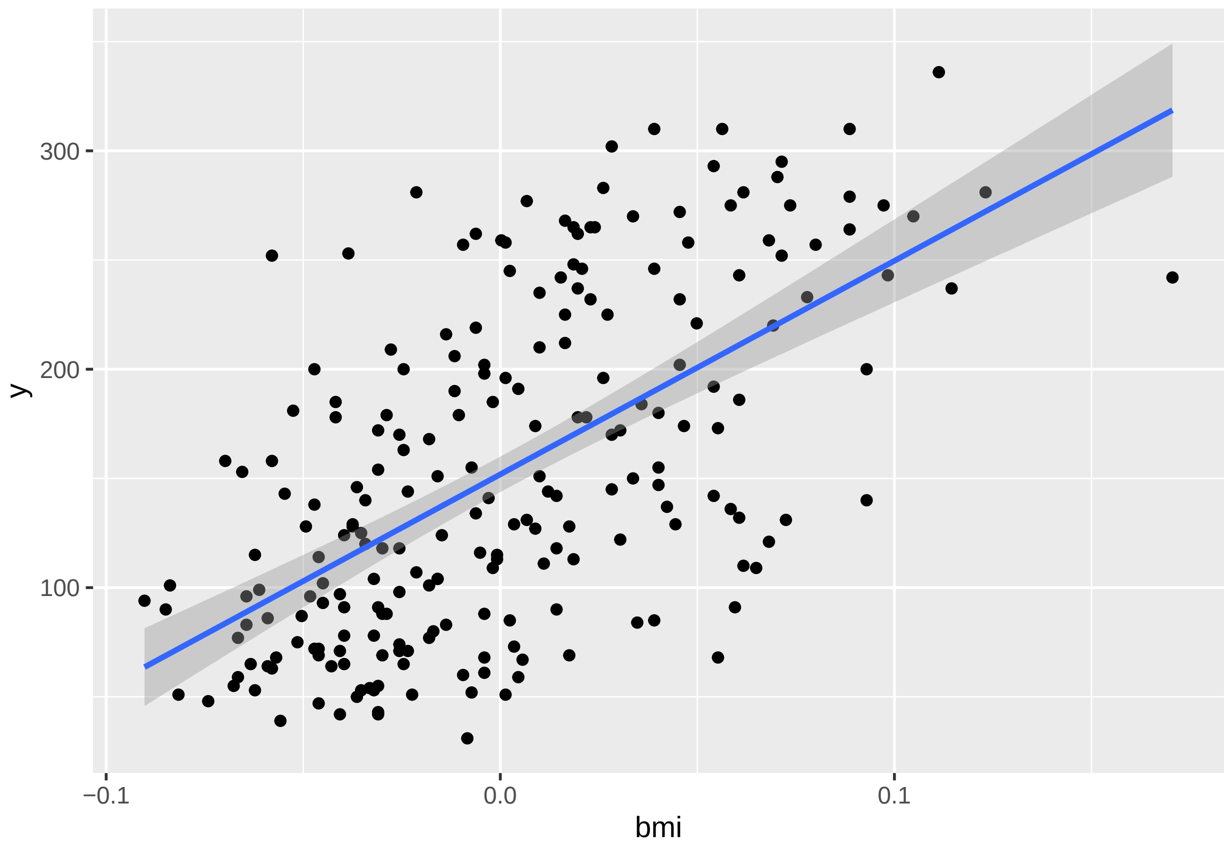
```
##
## Call:
## lm(formula = y ~ bmi, data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -137.860  -42.503   -8.696   46.538  156.788
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.862      4.119   36.87   <2e-16 ***
## bmi          977.720     88.060   11.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.22 on 219 degrees of freedom
## Multiple R-squared:  0.3602, Adjusted R-squared:  0.3572
## F-statistic: 123.3 on 1 and 219 DF,  p-value: < 2.2e-16
```

Scatter plot with regression line.

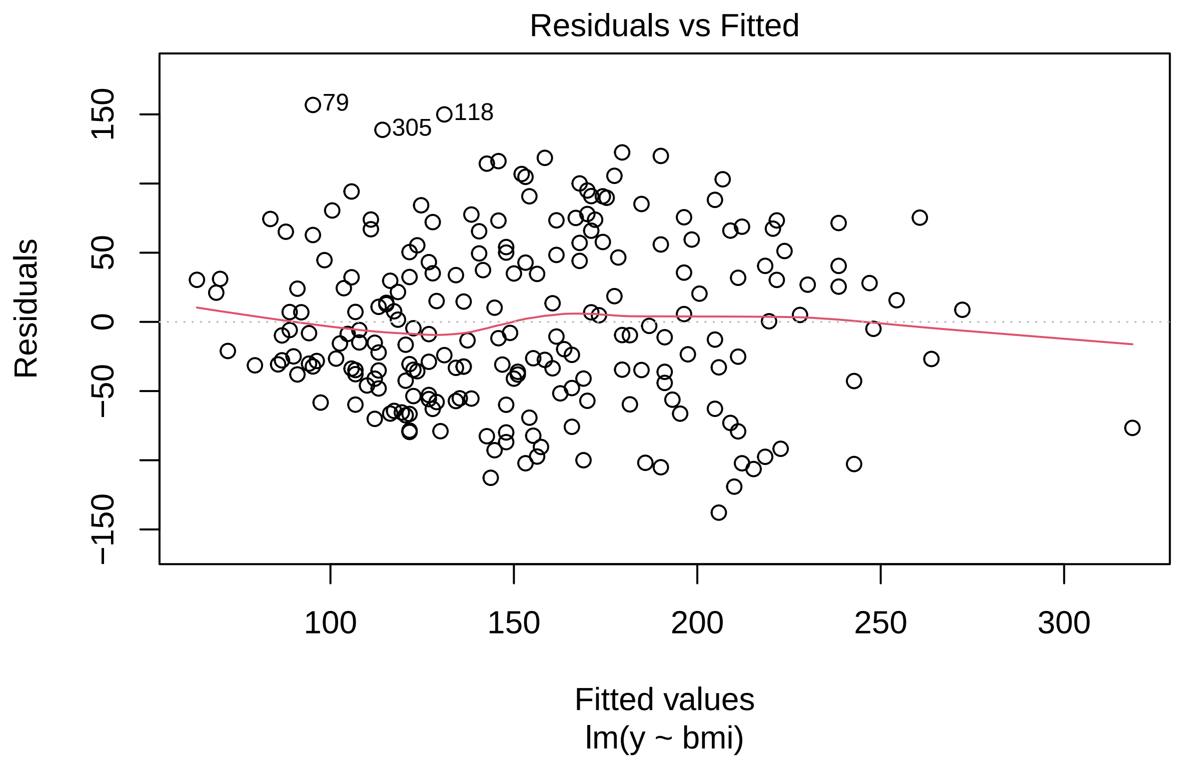
```
data_train%>%
  ggplot(data=.,aes(x=bmi,y=y))+
  geom_point()+
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



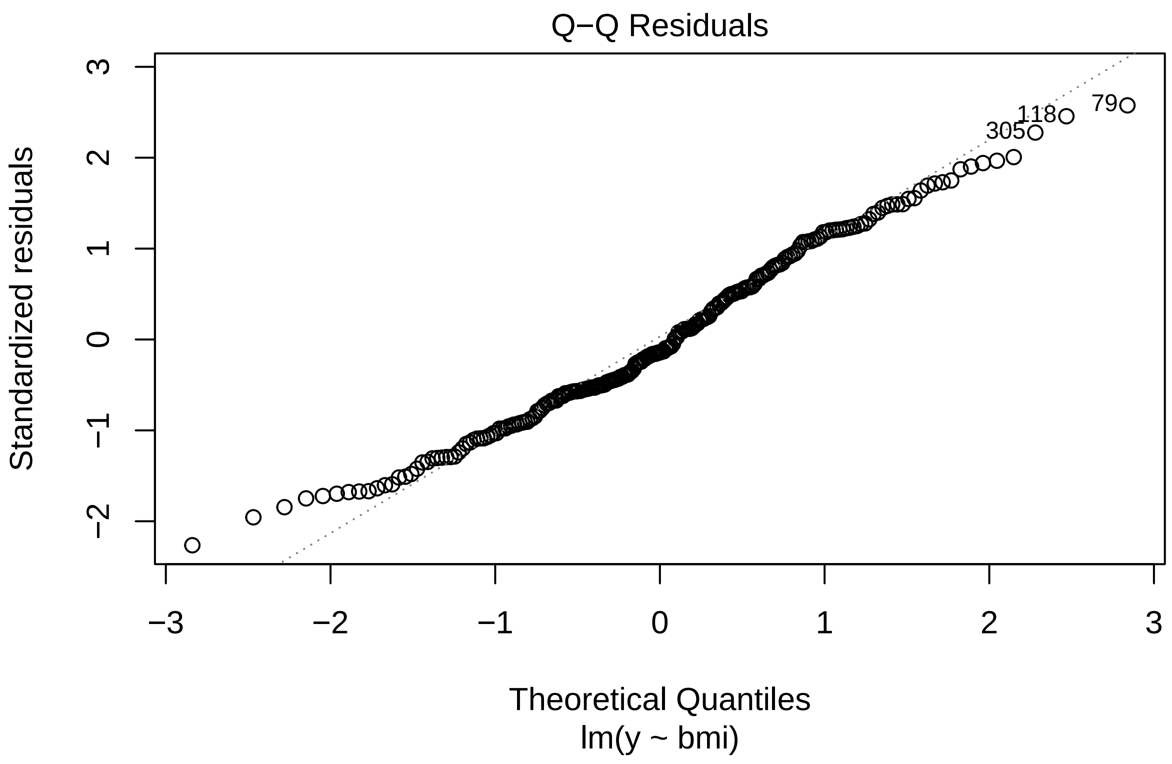
The Tukey Anscombe plot. The residuals scatter around the 0 line and do not show any systematic pattern. This indicates that the residuals are independent and have mean 0.

```
plot(fit1,which=1) # Tukey Anscombe plot
```

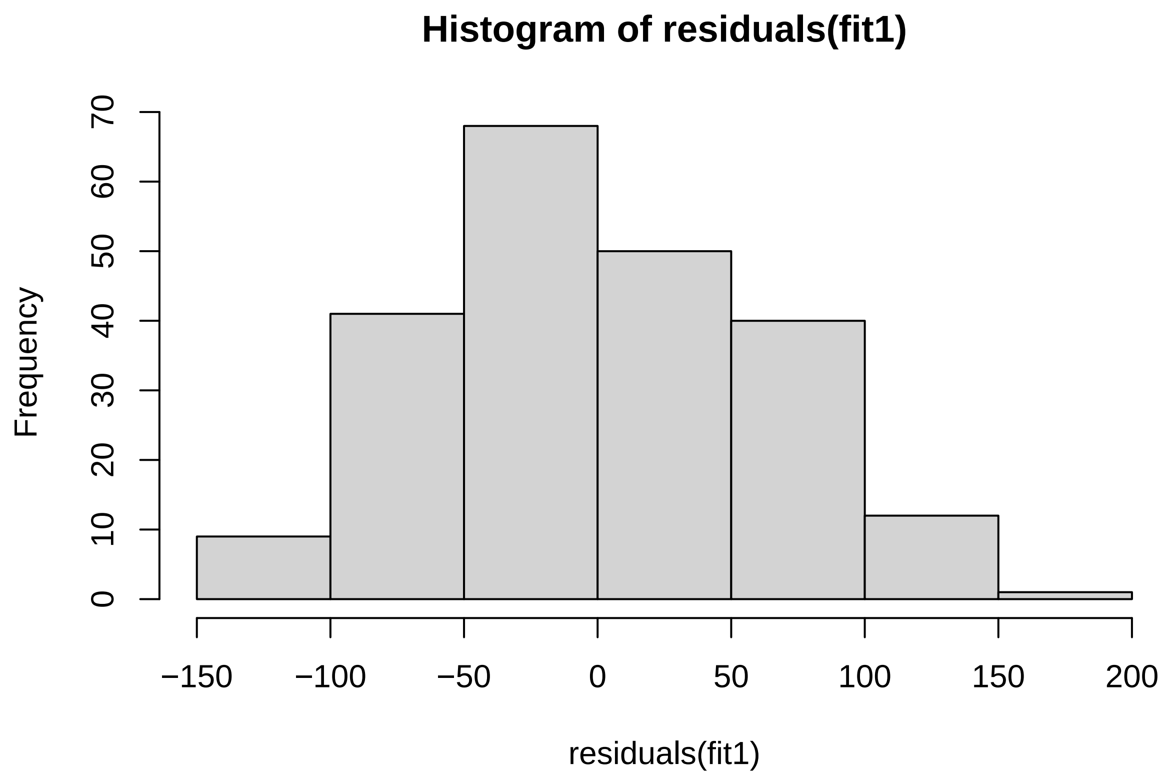


The QQ plot. This plot is used to check the normality assumption of the residuals. The residuals show slight tendency to be right-skewed (see also the histogram).

```r
plot(fit1,which=2) # QQ plot
```

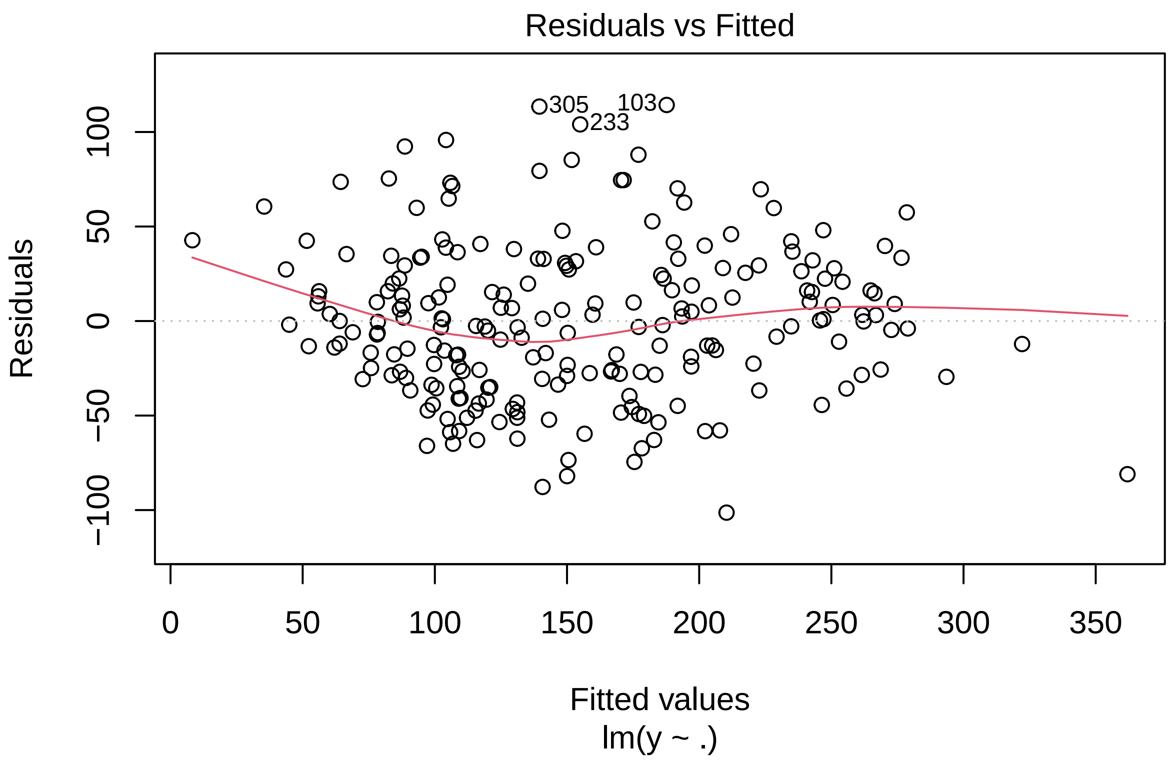


```r
hist(residuals(fit1))
```

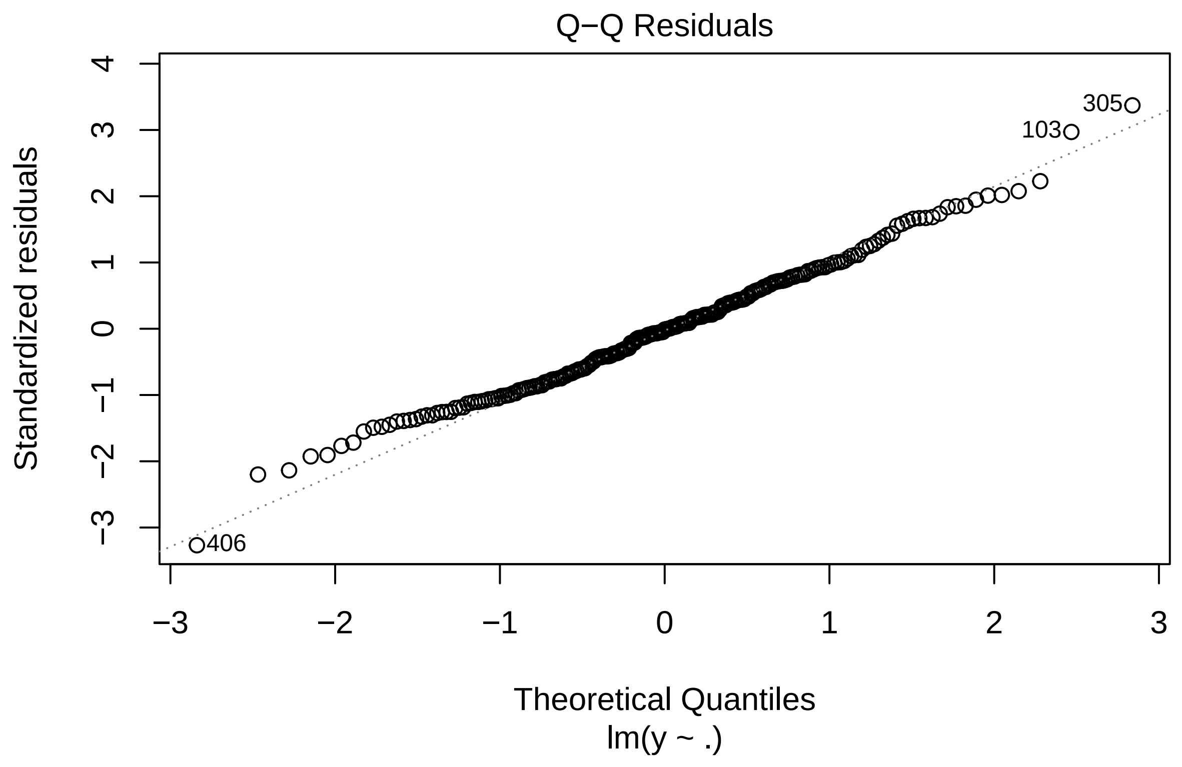


We run the multiple regression model with all covariates. We print the `summary` and create TA and QQ plots.

```
fit2 <- lm(y~.,data=data_train)
#summary(fit2)
plot(fit2,which=1)
```



```r
plot(fit2,which=2)
```



Calculate the RSS.

```r
sum(residuals(fit1)^2)
```

```
## [1] 820803.3
```

```r
sum(residuals(fit2)^2)
```

```
## [1] 366382
```

As expected the model with more covariates has a smaller RSS. We can compare the 2 models using the `anova` function. Based on the result we would favor the 2nd model.

```r
anova(fit1,fit2)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ bmi
## Model 2: y ~ age + sex + bmi + map + tc + ldl + hdl + tch + ltg + glu +
##     `age^2` + `bmi^2` + `map^2` + `tc^2` + `ldl^2` + `hdl^2` +
##     `tch^2` + `ltg^2` + `glu^2` + age.sex + age.bmi + age.map +
##     age.tc + age.ldl + age.hdl + age.tch + age.ltg + age.glu +
##     sex.bmi + sex.map + sex.tc + sex.ldl + sex.hdl + sex.tch +
##     sex.ltg + sex.glu + bmi.map + bmi.tc + bmi.ldl + bmi.hdl +
##     bmi.tch + bmi.ltg + bmi.glu + map.tc + map.ldl + map.hdl +
##     map.tch + map.ltg + map.glu + tc.ldl + tc.hdl + tc.tch +
##     tc.ltg + tc.glu + ldl.hdl + ldl.tch + ldl.ltg + ldl.glu +
##     hdl.tch + hdl.ltg + hdl.glu + tch.ltg + tch.glu + ltg.glu
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    219 820803
## 2    156 366382 63    454421 3.0712 8.344e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3 Diabetes data and model validation

In the previous section we developed 2 models to predict `y`. In this section we explore the generalizability of these models.

1. Calculate the RMSEs on the training data. Which model will perform better on future data?
2. Use the test data and make scatter plots of the observed against predicted outcomes. Use `ggplot` to create one plot per model and add the regression line (`geom_smooth`) and the "y=x" (`geom_abline`) line to the graph. This plot is also called "calibration plot". The model is "well" calibrated if the regression line agrees with the "y=x" line.
3. Generate boxplots of `predicted - observed` for the 2 models. What do you conclude?
4. Calculate the generalization error, i.e., the RMSE on the test data.

Solution to the exercise.

We calculate the RMSEs on the training data. RMSE on training data tells you how good the model "fits" the data. We cannot make any conclusion about the generalizability of the models based on RMSEs on training data. Model 2 includes many more covariates and therefore "fits" the data better (smaller training RMSE).

```r
library(caret) # RMSE is implemented in caret
RMSE(data_train$y,predict(fit1,newdata=data_train))
```

```
## [1] 60.94294
```

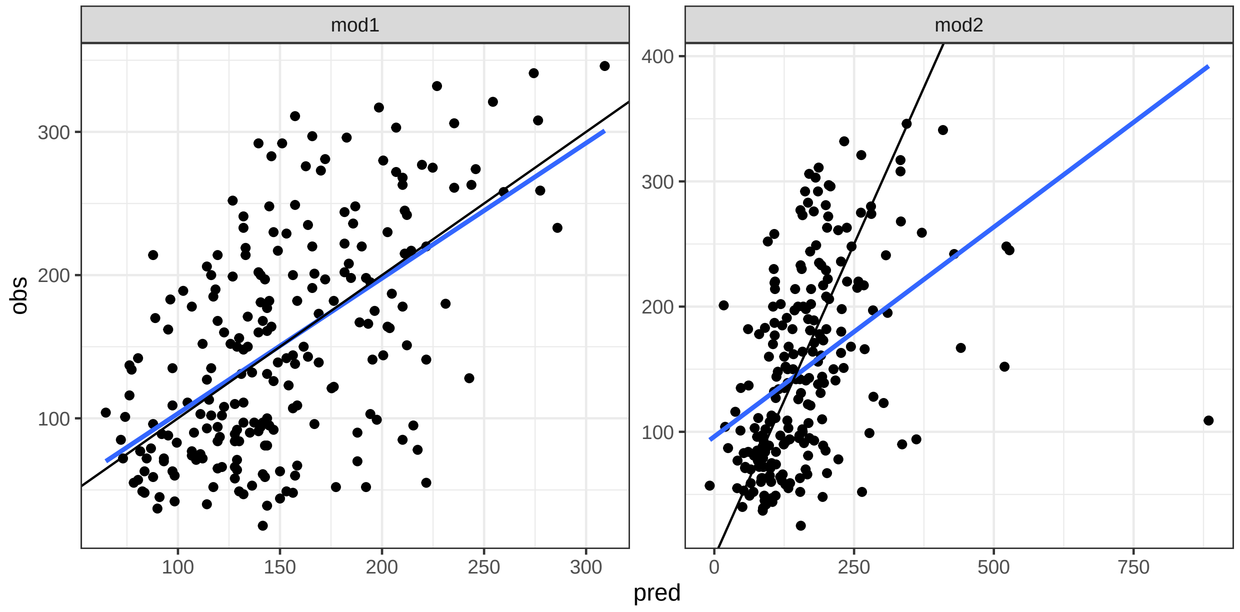
```r
RMSE(data_train$y,predict(fit2,newdata=data_train))
```

```
## [1] 40.71655
```

We draw calibration plots for the 2 models. Model 1 is better calibrated.

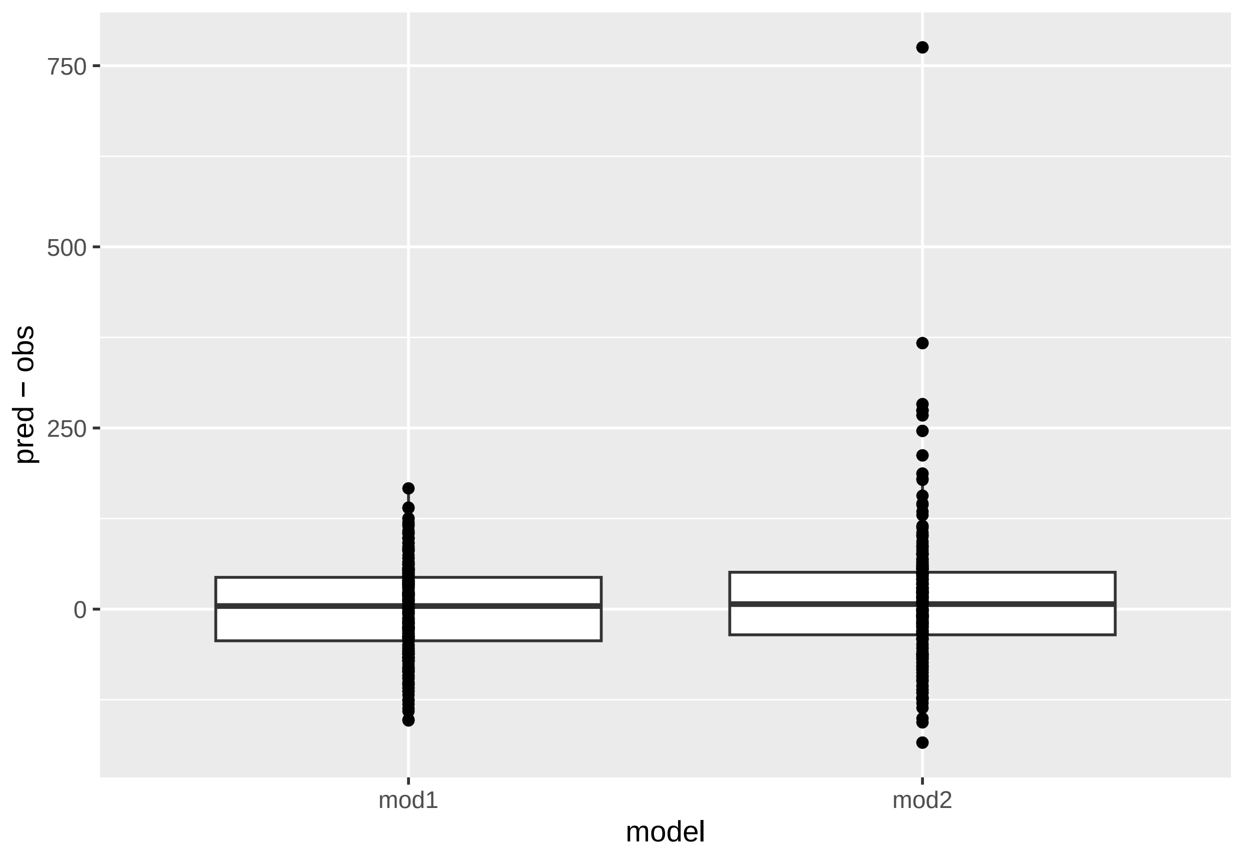
```r
dd <- data.frame(pred=c(predict(fit1,newdata=data_test),
                        predict(fit2,newdata=data_test)),
                 obs = rep(data_test$y,times=2),
                 model=rep(c("mod1","mod2"),each=nrow(data_test))
)
dd%>%
  ggplot(.,aes(x=pred,y=obs))+
  geom_point()+
  geom_smooth(se=FALSE,method="lm")+
  geom_abline(slope=1,intercept=0)+
  theme_bw()+
  facet_wrap(~model,scales="free")
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



Boxplots of predicted minus observed.

```
dd%>%
  ggplot(.,aes(x=model,y=pred-obs))+
  geom_boxplot()+
  geom_point()
```



Calculate RMSEs on test data. The RMSE on test data is smaller for model 1.

```
RMSE(data_test$y,predict(fit1,newdata=data_test))
```

```
## [1] 63.80156
```

```
RMSE(data_test$y,predict(fit2,newdata=data_test))
```

```
## [1] 97.87424
```

4 Calculus, optimization and OLS

1. Consider the function $f(x) = 2x^2 + x - 5$. Draw a plot of the function.
2. Use `optimize` to find the minimum of $f(x)$.
3. Obtain the minimum of $f(x)$ by taking the derivative and setting equal to zero.
4. Show that $\|a\|_2^2 = a^T a$.
5. Use the result in 4. and show that $\mathbf{RSS}(\beta) = \mathbf{y^T y} - \mathbf{2y^T X}\beta + \beta^{\mathbf{T}}\mathbf{X^T X}\beta$.
6. Invoke the result obtained in 4. and show that

$$\frac{\partial}{\partial\beta}\mathbf{RSS}(\beta) = -\mathbf{2X^T y} + \mathbf{2X^T X}\beta.$$

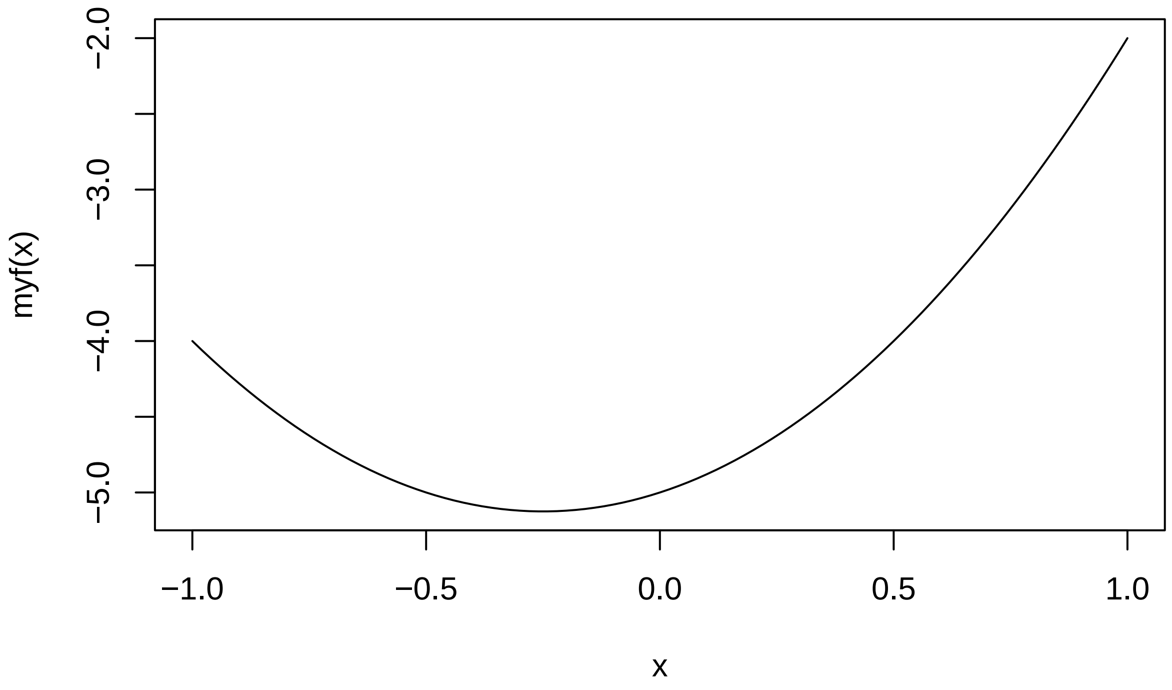
Hint: review the "Identities" section of Matrix calculus - Wikipedia.

7. Do you understand the derivation of the least squares estimator?

Solution to the exercise.

Plot of the function.

```
myf <- function(x){
  2*x^2 + x -5
}
curve(myf,from=-1,to=1)
```



```
optimize(myf,interval=c(-5,5))
```

```
## $minimum
## [1] -0.25
##
## $objective
## [1] -5.125
```

5 Diabetes data and regularization

The task is to use the diabetes data to construct a model that predicts the response `y` (disease progression) based on covariates. The two hopes are, that the model would produce accurate baseline predictions of response for future patients, and also that the form of the model would suggest which covariates were important factors in disease progression.

1. Read in the diabetes data set.
2. Run forward stepwise regression. Which is the first variable included in the selection process? Print the coefficients of the AIC-optimal model as a table.
3. Fit Ridge regression. Show the trace plot and the cross-validation plot.
4. Run the Lasso approach and show the trace and the cross-validation plots.
5. Calculate the root-mean-square errors (RMSE) for all 3 models on the test data and compare with the performance of the full model. Which model generalizes best?
6. Plot the regression coefficients for all 3 models.

The solution to this exercise.

Read the data set and create training and test data.

```
set.seed(007)
diabetes <- readRDS(file="data/diabetes.rds")
data <- as.data.frame(cbind(y=diabetes$y,diabetes$x2))
colnames(data) <- gsub(":",".",colnames(data))
train_ind <- sample(seq(nrow(data)),size=nrow(data)/2)
data_train <- data[train_ind,]
xtrain <- as.matrix(data_train[,-1])
ytrain <- data_train[,1]
data_test <- data[-train_ind,]
xtest <- as.matrix(data_test[,-1])
ytest <- data_test[,1]
```

We perform forward stepwise regression.

```
library(MASS) # stepAIC
```

```
# Full model
fit.full <- lm(y~.,data=data_train)

# Forward regression
fit.null <- lm(y~1,data=data_train)
fit.fw <- stepAIC(fit.null,direction="forward",
                  scope=list(lower=fit.null,
                             upper=fit.full

                  ),
                  trace = FALSE
)
#summary(fit.fw)
```

The selection process is depicted in the following table.

```
kable(as.data.frame(fit.fw$anova),digits=2,
      booktabs=TRUE)
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
	NA	NA	220	1295012.8	1919.37
+ ltg	1	480436.38	219	814576.4	1818.91

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
+ bmi	1	90618.04	218	723958.3	1794.85
+ map	1	39190.49	217	684767.8	1784.55
+ `glu^2`	1	32560.95	216	652206.9	1775.78
+ hdl	1	17629.79	215	634577.1	1771.72
+ sex	1	24729.23	214	609847.9	1764.94
+ age.sex	1	10928.14	213	598919.7	1762.94
+ sex.ldl	1	12011.61	212	586908.1	1760.47
+ glu	1	7553.66	211	579354.5	1759.60
+ bmi.map	1	6805.57	210	572548.9	1758.99
+ map.glu	1	10462.13	209	562086.8	1756.92

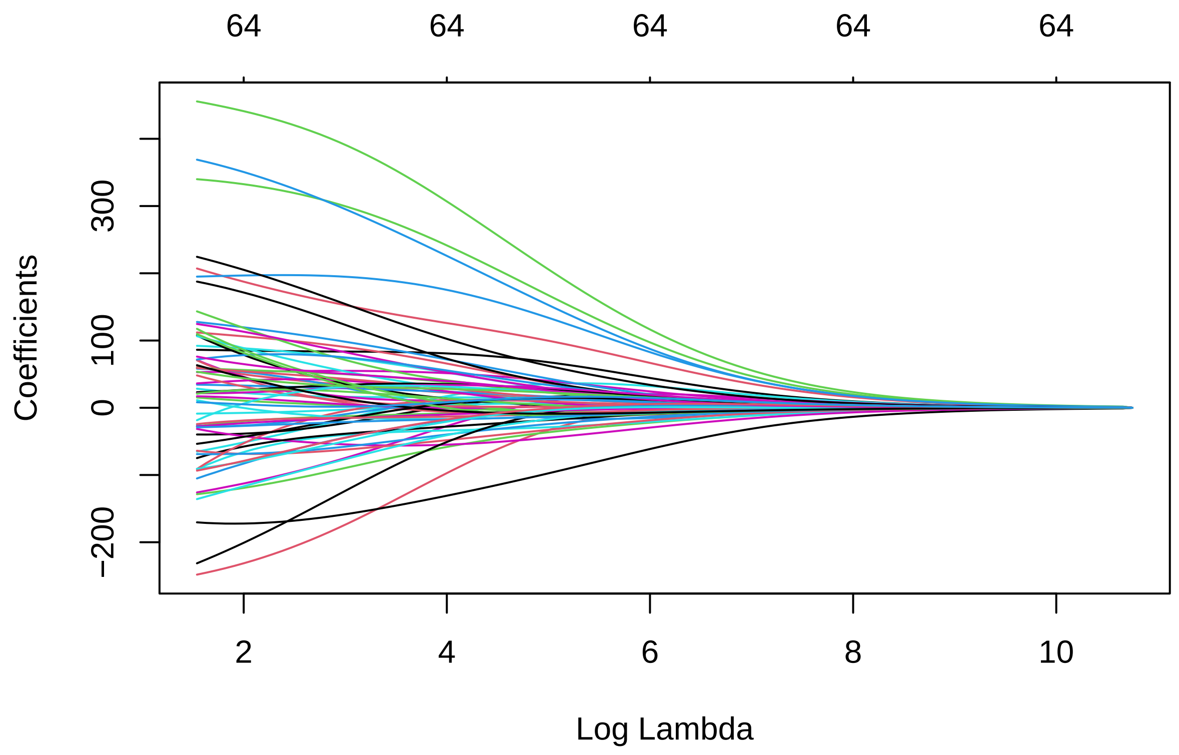
The regression coefficients and the corresponding statistics of the AIC-optimal model are shown next.

```r
kable(broom::tidy(fit.fw),digits=2,
      booktabs=TRUE)
```

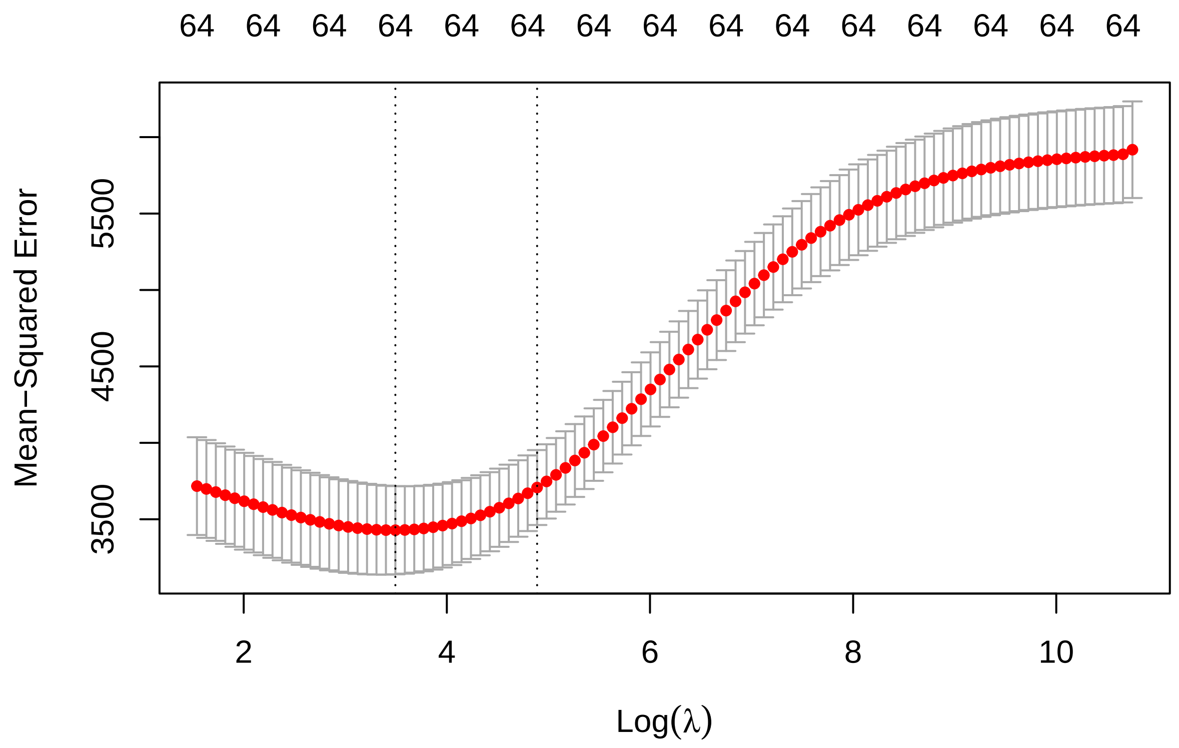
term	estimate	std.error	statistic	p.value
(Intercept)	152.74	3.54	43.13	0.00
ltg	506.51	97.36	5.20	0.00
bmi	351.03	90.31	3.89	0.00
map	399.67	90.07	4.44	0.00
`glu^2`	278.25	89.96	3.09	0.00
hdl	-271.94	85.48	-3.18	0.00
sex	-240.93	84.40	-2.85	0.00
age.sex	166.79	75.08	2.22	0.03
sex.ldl	-152.53	71.14	-2.14	0.03
glu	148.03	90.65	1.63	0.10
bmi.map	193.57	86.01	2.25	0.03
map.glu	-196.91	99.84	-1.97	0.05

We continue by fitting Ridge regression. We show the trace plot and the cross-validation plot.

```r
# Ridge
set.seed(1515)
library(glmnet)
fit.ridge <- glmnet(xtrain,ytrain,alpha=0)
fit.ridge.cv <- cv.glmnet(xtrain,ytrain,alpha=0)
plot(fit.ridge,xvar="lambda")
```

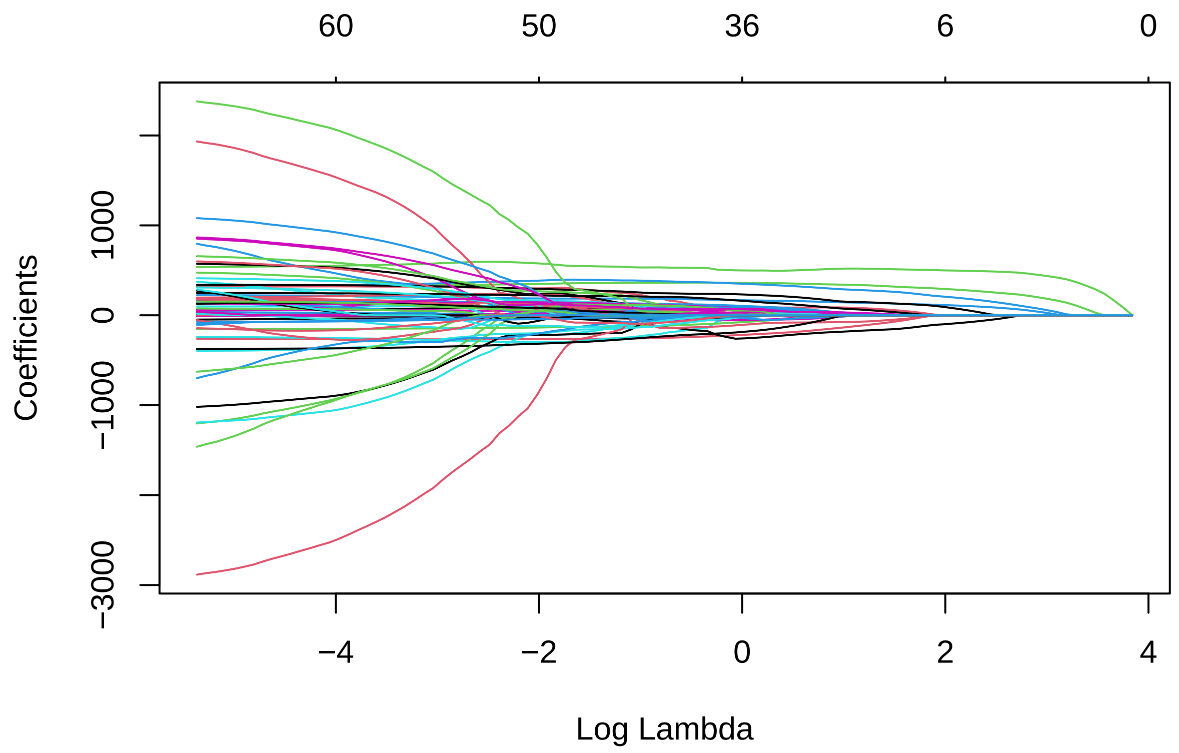


```r
plot(fit.ridge.cv)
```

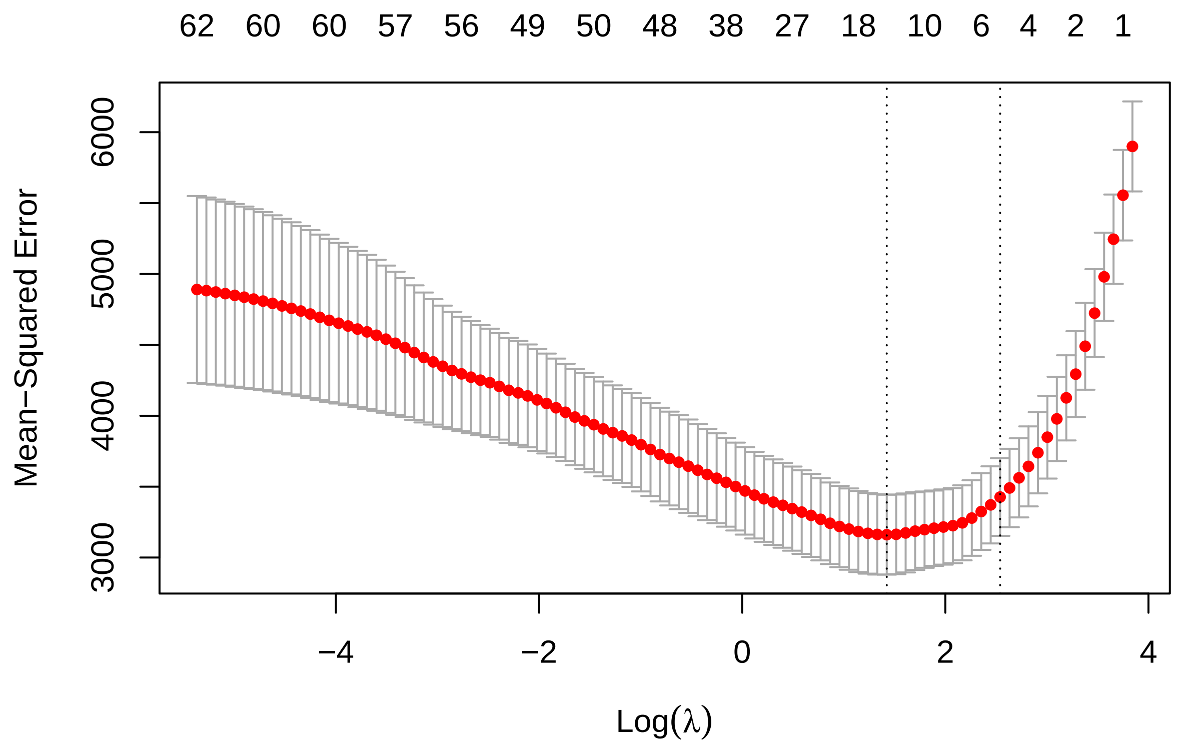


Finally, we run the Lasso approach and show the trace and the cross-validation plots.

```
# Lasso
set.seed(1515)
fit.lasso <- glmnet(xtrain,ytrain,alpha=1)
fit.lasso.cv <- cv.glmnet(xtrain,ytrain,alpha=1)
plot(fit.lasso,xvar="lambda")
```



```r
plot(fit.lasso.cv)#fit.lasso.cv$lambda.1se
```



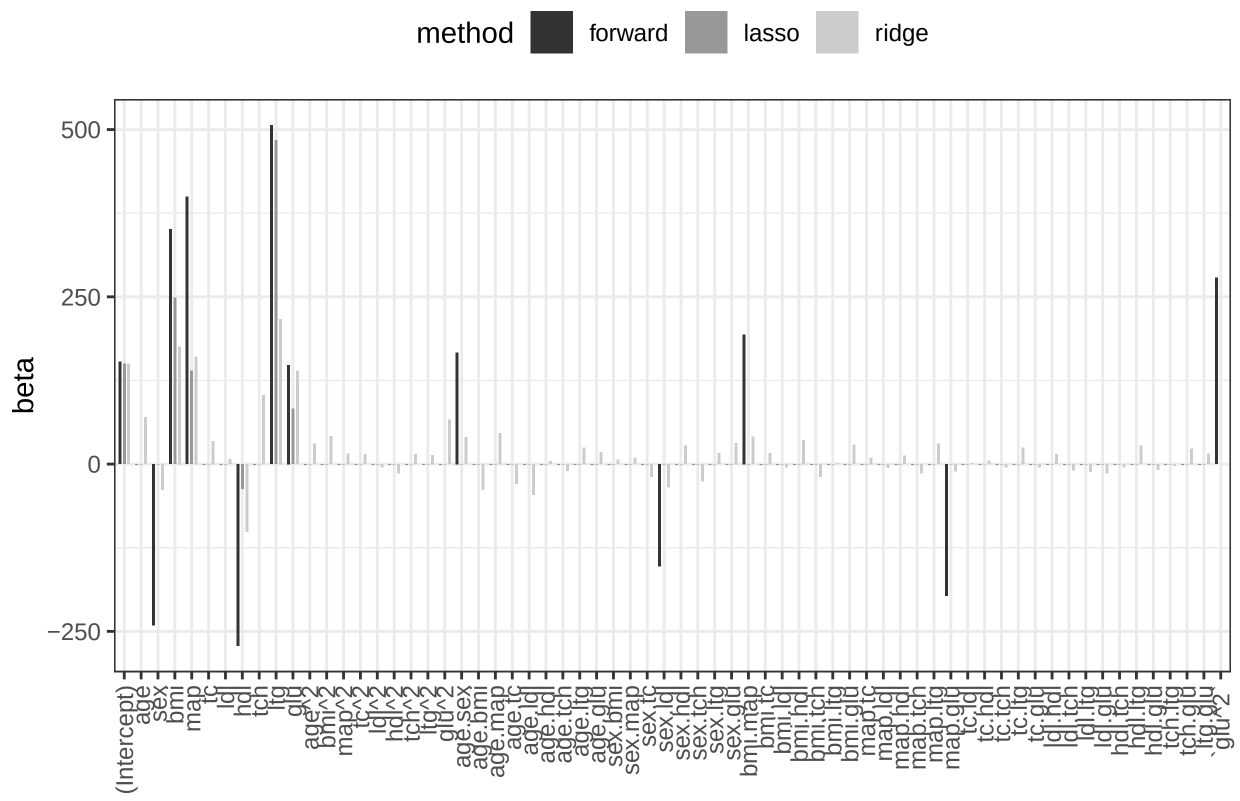
We calculate the root-mean-square errors (RMSE) on the test data and compare with the full model.

```
# RMSE
pred.full <- predict(fit.full,newdata=data_test)
pred.fw <- predict(fit.fw,newdata=data_test)
pred.ridge <- as.vector(predict(fit.ridge,newx=xtest,s=fit.ridge.cv$lambda.1se))
pred.lasso <- as.vector(predict(fit.lasso,newx=xtest,s=fit.lasso.cv$lambda.1se))
res.rmse <- data.frame(
  method=c("full","forward","ridge","lasso"),
  rmse=c(caret::RMSE(pred.full,ytest),caret::RMSE(pred.fw,ytest),caret::RMSE(pred.ridge,ytest),caret::R
kable(res.rmse,digits = 2,
      booktabs=TRUE)
```

method	rmse
full	81.90
forward	55.64
ridge	60.62
lasso	59.64

Forward stepwise regression has the lowest generalization error (RMSE). We plot the regression coefficients for all 3 methods.

```r
beta.fw <- coef(fit.fw)
beta.ridge <- coef(fit.ridge,s=fit.ridge.cv$lambda.1se)
beta.lasso <- coef(fit.lasso,s=fit.lasso.cv$lambda.1se)
res.coef <- data.frame(forward=0,ridge=as.numeric(beta.ridge),lasso=as.numeric(beta.lasso))
rownames(res.coef) <- rownames(beta.ridge)
res.coef[names(beta.fw),"forward"] <- beta.fw
res.coef$coef <- rownames(res.coef)
res.coef.l <- pivot_longer(res.coef,cols=c("forward","ridge","lasso"),names_to="method")

res.coef.l%>%
  dplyr::mutate(coef=factor(coef,levels = unique(coef)))%>%
  ggplot(.,aes(x=coef,y=value,fill=method))+
  geom_bar(width=0.5,position = position_dodge(width = 0.8),stat="identity")+
  theme_bw()+
  theme(legend.position = "top",axis.text.x = element_text(angle = 90,vjust = 0.5, hjust=1))+
  scale_fill_grey(aesthetics = c("fill","color"))+
  xlab("")+ylab("beta")
```



6 Diabetes data and the `caret` package

In the previous exercise we build predictive models for the Diabetes data. The `caret` package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The aim of this exercise is to get familiar with the `caret` package by running the code below and by investigating the purpose of the functions `train`, `trainControl`. Use the following link to learn more about `caret`.

```r
## Load package
library(caret)

## Read and prepare the data
set.seed(007)
diabetes <- readRDS(file="data/diabetes.rds")
data <- as.data.frame(cbind(y=diabetes$y,diabetes$x2))
colnames(data) <- gsub(":",".",colnames(data))
train_ind <- sample(seq(nrow(data)),size=nrow(data)/2)
data_train <- data[train_ind,]
xtrain <- as.matrix(data_train[,-1])
ytrain <- data_train[,1]
data_test <- data[-train_ind,]
xtest <- as.matrix(data_test[,-1])
ytest <- data_test[,1]

## Setup trainControl: 10-fold cross-validation
tc <- trainControl(method = "cv", number = 10)

## Ridge
lambda.grid <- fit.ridge.cv$lambda
fit.ridge.caret<-train(x=xtrain,
                       y=ytrain,
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 0,
                                              lambda=lambda.grid),
                       trControl = tc
)

# CV curve
plot(fit.ridge.caret)
# Best lambda
fit.ridge.caret$bestTune$lambda
# Model coefficients
coef(fit.ridge.caret$finalModel,fit.ridge.cv$lambda.1se)%>%head
# Make predictions
fit.ridge.caret %>% predict(xtest,s=fit.ridge.cv$lambda.1se)%>%head

## Lasso
lambda.grid <- fit.lasso.cv$lambda
fit.lasso.caret<-train(x=xtrain,
                       y=ytrain,
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 1,
                                              lambda=lambda.grid),
                       trControl = tc
)

# CV curve
plot(fit.lasso.caret)
# Best lambda
```

```r
fit.lasso.caret$bestTune$lambda
# Model coefficients
coef(fit.lasso.caret$finalModel,
     fit.lasso.caret$bestTune$lambda)%>%head
# Make predictions
fit.lasso.caret%>%predict(xtest,
                          s=fit.ridge.cv$lambda.1se)%>%head

## Compare Ridge and Lasso
models <- list(ridge=fit.ridge.caret,lasso=fit.lasso.caret)
resamples(models) %>% summary( metric = "RMSE")
```

7 Closed form solution for Ridge regression

1. Show that the Ridge optimization problem has the closed form solution

$$\hat{\beta}_{\lambda}^{\text{Ridge}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y}.$$

Hint: calculate the gradient of the loss function $\ell_{\text{Ridge}}(\beta|\mathbf{y},\mathbf{X}) = \text{RSS}(\beta) + \lambda\|\beta\|_2^2$, set equal to zero and solve for β.

2. Use the code below to generate simulated data. Use the formula from the script to calculate the Ridge coefficients for $\lambda = 35$. Compare the coefficients with those obtained using `glmnet`. Hint: Read the following blog on how to scale the λ.

```r
set.seed(1)

# simulate data
n <- 20
p <- 15
x <-  matrix(rnorm(n * p), n, p)
y <- x[,1:4]%*%c(2,-2,2,-2)+rnorm(n)
```

Solution to the exercise.

<u>Closed form solution for Ridge regression</u>

$$\hat{\beta}_c^{\text{Ridge}} = \underset{\|\beta\|_2^2 \le c}{\operatorname{argmin}} \ \|y - X\beta\|_2^2$$

Lagrange multiplier

$$\Longleftrightarrow \quad \text{minimize} \quad \underbrace{\|y-X\beta\|_2^2 + \lambda\|\beta\|_2^2}_{\equiv\, \ell(\beta)}$$

$$\Longleftrightarrow \quad \frac{\partial \ell(\beta)}{\partial \beta} \overset{!}{=} 0$$

$$\|y-X\beta\|_2^2 = (y-X\beta)^T(y-X\beta) = \underset{\substack{\downarrow \frac{\partial}{\partial\beta} \\ 0}}{y^Ty} - \underset{\substack{\downarrow \frac{\partial}{\partial\beta} \\ X^Ty}}{y^TX\beta} - \underset{\substack{\downarrow \frac{\partial}{\partial\beta} \\ X^Ty}}{\beta^TX^Ty} + \underset{\substack{\downarrow \\ 2X^TX\beta}}{\beta^TX^TX\beta}$$

$$\Rightarrow \frac{\partial}{\partial\beta}\|y-X\beta\|_2^2 = \underline{-2X^Ty + 2X^TX\beta}$$

$$\|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2 \qquad \Rightarrow \frac{\partial}{\partial\beta}\|\beta\|_2^2 = \underline{2\beta}$$

$$\Rightarrow \frac{\partial \ell(\beta)}{\partial \beta} = -2X^Ty + 2X^TX\cdot\beta + \lambda 2\beta \overset{!}{=} 0$$

$$\Longleftrightarrow \quad X^TX\beta + \lambda\beta \overset{!}{=} X^Ty$$

$$\Longleftrightarrow \quad (X^TX + \lambda I)\,\beta \overset{!}{=} X^Ty$$

$$\Longleftrightarrow \quad \hat{\beta} = (X^TX + \lambda I)^{-1} X^Ty$$

$\boxtimes$

We obtain the Ridge coefficients using `glmnet`.

```
my.lambda <- 35
fit.ridge <- glmnet(x,y,alpha=0,lambda=my.lambda,
                    intercept=FALSE,standardize = FALSE,thresh = 1e-20,exact=TRUE)
coef.ridge <- as.vector(coef(fit.ridge))[-1]
head(coef.ridge)
```

```
## [1]  0.27330465 -0.24799766  0.19686435 -0.21942808  0.05302251 -0.02458886
```

Next we calculate the coefficients based on the formula from the script. Note that we need to re-scale the lambda.

```
sd_y <- sqrt(var(y)*(n-1)/n)[1,1]
my.lambda2 <- n*my.lambda/sd_y
coef.ridge2 <- solve(t(x)%*%x+my.lambda2*diag(nrow=ncol(x)))%*%t(x)%*%y
head(coef.ridge2)[,1]
```

```
## [1]  0.27031012 -0.24528806  0.19469189 -0.21696366  0.05244081 -0.02429057
```

8 Bayesian interpretation of Ridge regression (difficult)

1. Write down the log-likelihood of the linear regression model. Note: $Y_i = X_i^T\beta + \epsilon_i$, where $\epsilon_1, \ldots, \epsilon_n$ iid $N(0, \sigma^2)$ and $\mathbf{X}$ is a fixed $n \times p$ design matrix.
2. Find the expression for the maximum likelihood estimator.
3. Assuming a prior distribution $\beta_1, \ldots, \beta_p$ iid $\sim N(0, \tau^2)$, derive the posterior distribution of β and show that the maximum a posteriori estimator (MAP) coincides with the Ridge estimator.

The solution to this exercise.

$\boxed{\text{Likelihood of linear regression}}$

$$p(y_i | x_i, \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_i - x_i^T\beta)^2}{2\sigma^2} \right)$$

$$\log p(y_i | x_i, \beta, \sigma^2) \propto -\log \sigma - \frac{(y_i - x_i^T\beta)^2}{2\sigma^2}$$

$$\log p(Y|X, \beta, \sigma^2) = \sum_{i=1}^{n} \log p(y_i | x_i, \beta, \sigma^2) \propto -n \log \sigma - \frac{1}{2\sigma^2} \|Y - X\beta\|_2^2$$

$\boxed{\text{Maximum Likelihood for linear regression}}$

A) $\max_\beta \log p(Y|X, \beta, \sigma^2) \Leftrightarrow \min_\beta \|Y - X\beta\|_2^2 \Leftrightarrow \hat{\beta}^{OLS}$

B) $\max_\sigma \log p(Y|X, \beta, \sigma^2) \Leftrightarrow \frac{\partial}{\partial\sigma} \log p(Y|X, \beta, \sigma^2) \overset{!}{=} 0$

$$\frac{\partial}{\partial\sigma} \log p(Y|X, \hat{\beta}^{OLS}, \sigma^2) = \frac{-n}{\sigma} + \frac{1}{\sigma^3} \|Y - X\hat{\beta}^{OLS}\|^2 \overset{!}{=} 0$$

$$\hat{\sigma}^2 = \frac{\|Y - X\hat{\beta}^{OLS}\|^2}{n}$$

$\boxed{\text{Posteriori distribution}}$

$$p(\beta | Y, X) = \frac{p(Y|X, \beta) \; p(\beta)}{p(Y|X)}$$

$$\log p(\beta | Y, X) \propto \log p(Y|X, \beta) + \overbrace{\log p(\beta)}^{\propto -p \log \tau - \frac{1}{2\tau^2} \|\beta\|_2^2}$$

$$\propto -\cancel{n \log \sigma} - \frac{1}{2\sigma^2} \|Y - X\beta\|_2^2 - \cancel{p \log \tau} - \frac{1}{2\tau^2} \|\beta\|_2^2$$

$$\rightsquigarrow \max_\beta \log p(\beta | Y, X) \Leftrightarrow \min_\beta \|Y - X\beta\|_2^2 + \underbrace{\frac{\sigma^2}{\tau^2}}_{= \lambda} \|\beta\|_2^2$$

---

9 Riboflavin data and elasticnet mixing parameter

1. Load the `hdi` package and load the riboflavin data set `riboflavin.rds`. More information on the data can be obtained using `?riboflavin`.
2. Run the Lasso and generate the trace plot.
3. Run the Elastic net with mixing parameters $\alpha = 0.25, 0.5, 0.75$ and 1 and compare the cross-validation curves. Hint: use the `foldid` argument in `glmnet`.
4. Show the selected genes for the best performing model.

The solution to this exercise.

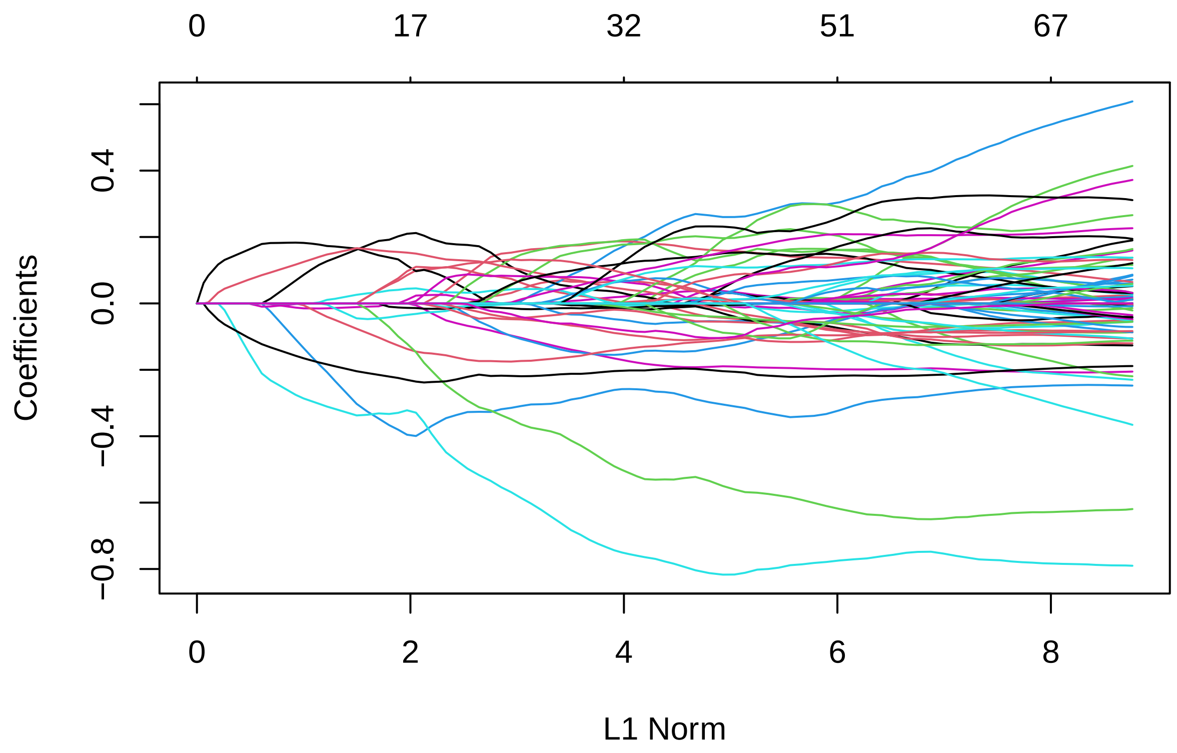
We first load the data and check the data structure.

```
library(hdi)
library(glmnet)
riboflavin <- readRDS(file="data/riboflavin.rds")
str(riboflavin)
```

```
## 'data.frame':    71 obs. of  2 variables:
##  $ y: num  -6.64 -6.95 -7.93 -8.29 -7.31 ...
##  $ x: 'AsIs' num [1:71, 1:4088] 8.49 7.64 8.09 7.89 6.81 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:71] "b_Fbat107PT24.CEL" "b_Fbat107PT30.CEL" "b_Fbat107PT48.CEL" "b_Fbat107PT52.CEL
##   .. ..$ : chr [1:4088] "AADK_at" "AAPA_at" "ABFA_at" "ABH_at" ...
```

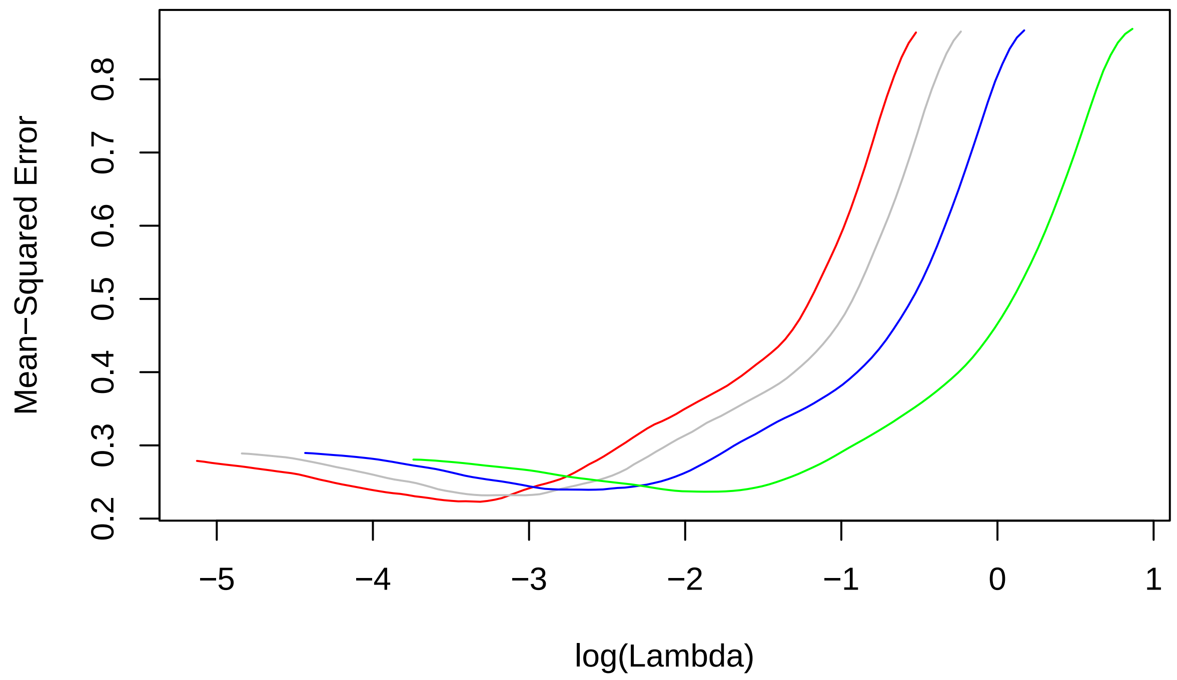
Next we setup the design matrix and the response variable and we run the Lasso.

```
x <- riboflavin[,-1]
y <- riboflavin[,1]
fit <- glmnet(x = x, y = y)
plot(fit)
```



We run 10-fold cross-validation for the different mixing parameters and plot the error curves.

```
set.seed(1)
n.fold <- 10
foldid <- sample(rep(1:n.fold,length=length(y)))
cv1 <- cv.glmnet(x, y, foldid = foldid, alpha = 1)
cv2  <- cv.glmnet(x, y, foldid = foldid, alpha = 0.75)
cv3  <- cv.glmnet(x, y, foldid = foldid, alpha = 0.5)
cv4  <- cv.glmnet(x, y, foldid = foldid, alpha = 0.25)

t.lambdarange <- range(log(c(cv1$lambda,
                             cv2$lambda,
                             cv3$lambda,
                             cv4$lambda)))
t.crange <- range(c(cv1$cvm,cv2$cvm,cv3$cvm,cv4$cvm))
plot(log(cv1$lambda), cv1$cvm ,
     pch = 19, col = "red",
     xlab = "log(Lambda)",
     ylab=cv1$name,
     type="l",
     xlim=t.lambdarange,
     ylim=t.crange)
lines(log(cv2$lambda), cv2$cvm, pch = 19, col = "grey")
lines(log(cv3$lambda) , cv3$cvm , pch = 19, col = "blue")
lines(log(cv4$lambda) , cv4$cvm , pch = 19, col = "green")
```



Finally, we print the gene names of the non-zero coefficients.

```
## Get selected genes
b <- as.matrix(coef(cv1))
rownames(b)[b!=0]
```

```
##  [1] "(Intercept)" "ARGF_at"     "DNAJ_at"     "GAPB_at"     "LYSC_at"     "PCKA_at"     "PKSA_at"
##  [9] "SPOVAA_at"   "XHLB_at"     "XKDS_at"     "XTRA_at"     "YBFI_at"     "YCGO_at"     "YCKE_at"
## [17] "YCLF_at"     "YDDK_at"     "YEBC_at"     "YEZB_at"     "YFHE_r_at"   "YFIR_at"     "YHDS_r_at"
## [25] "YRVJ_at"     "YURQ_at"     "YXLD_at"     "YXLE_at"     "YYDA_at"
```

```
## By default, the selected variables are based on the largest value of
## lambda such that the cv-error is within 1 standard error of the minimum
```

10 Ridge and Lasso for the orthonormal design (difficult)

1. Calculate the Ridge and the Lasso solution for the special case of an orthonormal design matrix.

The solution to this exercise.

Ridge and Lasso for orthonormal design

orthonormal design $\Rightarrow$ $\boxed{X^T X = I}$

$\hat{\beta}^{OLS} = (X^T X)^{-1} X^T y = X^T y$ Note: $\hat{\beta}^{OLS\,T} \hat{\beta}^{OLS} = y^T X X^T y$

$$\| y - X\beta \|_2^2 = (y - X\beta)^T (y - X\beta) ::$$

$$= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X \beta$$

$$= y^T y - 2\beta^T \underbrace{X^T y}_{\hat{\beta}^{OLS}} + \beta^T \beta$$

$$\propto -2\beta^T \hat{\beta}^{OLS} + \beta^T \beta$$

Ridge: $\frac{\partial}{\partial \beta}\left(-2\beta_j \hat{\beta}_j^{OLS} + \beta_j^2 + \lambda \beta_j^2\right) \overset{!}{=} 0$

$$-2\hat{\beta}_j^{OLS} + 2\beta_j + 2\lambda\beta_j \overset{!}{=} 0$$

$$\Rightarrow (1+\lambda)\beta_j \overset{!}{=} \hat{\beta}_j^{OLS}$$

$$\Rightarrow \underline{\hat{\beta}_j^{Ridge} \overset{!}{=} \frac{1}{1+\lambda} \hat{\beta}_j^{OLS}}$$

Lasso: $\frac{\partial}{\partial \beta}\left(-2\beta_j \hat{\beta}_j^{OLS} + \beta_j^2 + \lambda |\beta_j|\right) \overset{!}{=} 0$

If $\beta_j \neq 0$ $\quad -\hat{\beta}_j^{OLS} + \beta_j + \frac{\lambda}{2}\,\mathrm{sgn}(\beta_j) \overset{!}{=} 0$

$$\Rightarrow \hat{\beta}_j^{Ridge} = \mathrm{sgn}(\hat{\beta}_j^{OLS}) \left(|\hat{\beta}_j^{OLS}| - 0.5\lambda \right)_+$$

11 Heart disease data and logistic regression

We explore logistic regression based on the South African heart disease data `sahd.rds`. Proceed as follows:

1. Fit a univariate logistic regression model with `age` as the covariate. Calculate the odds-ratio and elaborate on the interpretation.
2. Predict the probability of heart disease at age 65.

3. Fit a logistic regression model including all covariates. Run stepwise backward selection. Which variables are excluded? What is the AIC value of the final model?
4. Fit a logistic regression model using four natural cubic spline bases for each term in the model. Run backward selection and summarise the final model. Plot the natural cubic spline functions for the age term (use `termplot`). What does the plot tell you?

The solution to this exercise.

We load the data and fit a logistic regression with age as the covariate.

```r
sahd <- readRDS(file="data/sahd.rds")
fit <- glm(chd~age, data=sahd, family=binomial )
summary(fit)
```

```
##
## Call:
## glm(formula = chd ~ age, family = binomial, data = sahd)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.521710   0.416031  -8.465  < 2e-16 ***
## age          0.064108   0.008532   7.513 5.76e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 596.11  on 461  degrees of freedom
## Residual deviance: 525.56  on 460  degrees of freedom
## AIC: 529.56
##
## Number of Fisher Scoring iterations: 4
```

The odds ratio for age is given next.

```r
exp(coef(fit)[2])
```

```
##      age
## 1.066208
```

This means that an increase of 1 year in age leads to a 6.6% increase in the odds of having a heart disease.

The estimated probability of having a heart disease at age 65 can be calculated using the `predict` function.

```
predict(fit,newdata=data.frame(age=65),type="response")
```

```
##         1
## 0.6559532
```

Alternatively, we can use the inverse logit formula.

```
lp <- coef(fit)[1]+coef(fit)[2]*65
exp(lp)/(exp(lp)+1)
```

```
## (Intercept)
##   0.6559532
```

We fit a logistic regression model including all covariates. Then we perform stepwise backward selection using `stepAIC`.

```
fit.full <- glm(chd~sbp+tobacco+ldl+famhist+obesity+alcohol+age,
                    data=sahd,
                    family="binomial")
fit.bw <- stepAIC(fit.full,direction = "backward",trace=FALSE)
```

The terms removed in each step are provided in the next table.

```
kable(as.data.frame(fit.bw$anova),digits=3,booktabs=TRUE)
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
	NA	NA	454	483.174	499.174
- alcohol	1	0.019	455	483.193	497.193
- sbp	1	1.104	456	484.297	496.297
- obesity	1	1.147	457	485.444	495.444

The variables alcohol, sbp and obesity are excluded from the model. The AIC values are provided in the table above. We can also re-calculate the AIC for the final model.

```
AIC(fit.bw)
```

```
## [1] 495.4439
```

We fit a logistic regression model using natural splines.

```
library(splines) # to fit splines
```

```
# Computes the logistic regression model using natural splines (note famhist is included as a factor):
form <-  "chd ~ ns(sbp,df=4) + ns(tobacco,df=4) + ns(ldl,df=4) + famhist + ns(obesity,df=4)+ ns(alcohol
form <-  formula(form)
fit <-  glm( form, data=sahd, family=binomial )

# stepwise backward selection
fit.bw <- stepAIC(fit,direction="backward",trace = 0)
kable(as.data.frame(fit.bw$anova),digits=3,booktabs=TRUE)
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
	NA	NA	436	457.632	509.632
- ns(alcohol, df = 4)	4	0.456	440	458.088	502.088

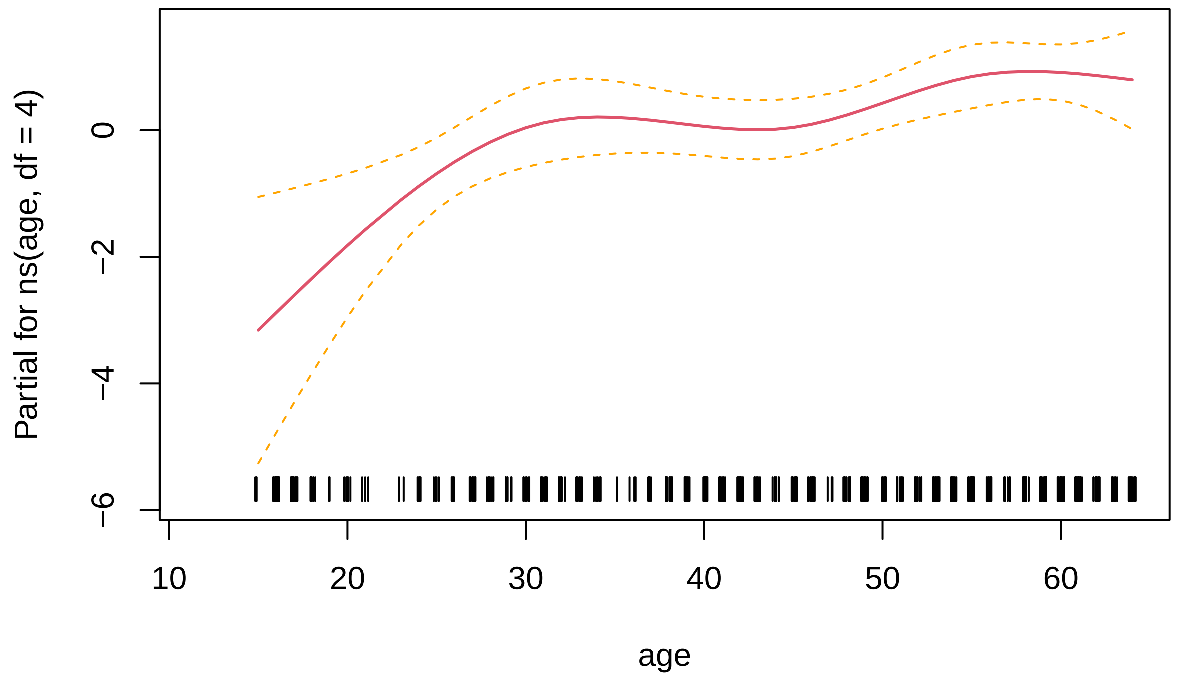
The summary of the final model is provided next.

```
kable(as.data.frame(drop1(fit.bw, test="Chisq" )),digits=2,booktabs=TRUE)
```

	Df	Deviance	AIC	LRT	Pr(>Chi)
	NA	458.09	502.09	NA	NA
ns(sbp, df = 4)	4	467.16	503.16	9.08	0.06
ns(tobacco, df = 4)	4	470.48	506.48	12.39	0.01
ns(ldl, df = 4)	4	472.39	508.39	14.31	0.01
famhist	1	479.44	521.44	21.36	0.00
ns(obesity, df = 4)	4	466.24	502.24	8.15	0.09
ns(age, df = 4)	4	481.86	517.86	23.77	0.00

We can plot the natural spline function for the first term as follows.

```
termplot(fit.bw,se=TRUE,rug=TRUE,term=6)
```



The plot shows how the log-odds change with age (keeping the other variables fixed). We observe a slight deviation from linearity, i.e. the log-odds increase more strongly for age <35 than for age >35.

12 Phoneme recognition

In this exercise we investigate prediction of phonemes based on digitized speech data.

1. Read the data set, subset the phonemes "aa" and "ao" and create training and test data.

```r
dat <- readRDS(file="data/phoneme.rds")
dat2 <- dat[dat$g%in%c("aa","ao"),]

dtrain <- dat2[grepl("^train",dat2$speaker),-c(1,259)]
xtrain <- as.matrix(dtrain[,-257])
ytrain <- ifelse(dtrain$g=="ao",1,0)
dtest <- dat2[grepl("^test",dat2$speaker),-c(1,259)]
xtest <- as.matrix(dtest[,-257])
ytest <- ifelse(dtest$g=="ao",1,0)

dtrain$y <- ytrain
dtest$y <- ytest
dtrain <- dtrain[,-257]
dtest <- dtest[,-257]
```

2. Plot the log-periodogram as a function of frequency for 5 examples each of the phonemes "aa" and "ao".
3. Fit a logistic regression model and evaluate the training and test misclassification errors.
4. Run Lasso regression and evaluate the training and test misclassification errors.
5. In the previous approaches we assumed logit-link

$$\text{logit}(x;\beta) = \sum_{j=1}^{256} X_j\beta_j.$$

Next we assume that the coefficients are a smooth function of the frequency $\beta(f)$, i.e.

$$\beta(f) = \sum_{m=1}^{\nu} h_m(f)\theta_m,$$

where h_m are B-spline basis functions for a natural cubic spline with $\nu = 12$ degrees of freedom (defined on the set of frequencies). Consider filtered inputs $x^* = \mathbf{H}^T x$ and fit θ by logistic regression on the x^*. Evaluate the training and test misclassification errors.

6. Plot the coefficients of the different models.

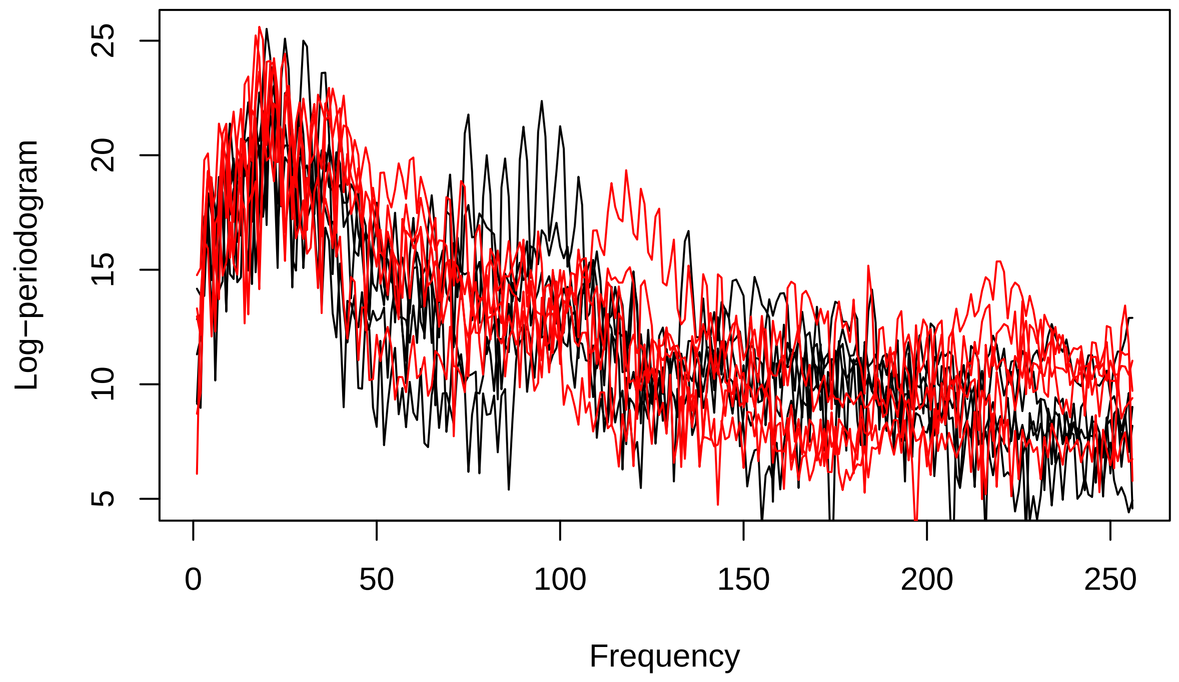
The solution to this exercise.

We prepare the data set.

```r
dat <- readRDS(file="data/phoneme.rds")
dat2 <- dat[dat$g%in%c("aa","ao"),]

dtrain <- dat2[grepl("^train",dat2$speaker),-c(1,259)]
xtrain <- as.matrix(dtrain[,-257])
ytrain <- ifelse(dtrain$g=="ao",1,0)
dtest <- dat2[grepl("^test",dat2$speaker),-c(1,259)]
xtest <- as.matrix(dtest[,-257])
ytest <- ifelse(dtest$g=="ao",1,0)

dtrain$y <- ytrain
dtest$y <- ytest
dtrain <- dtrain[,-257]
dtest <- dtest[,-257]
```

We plot the log-periodograms.

```r
set.seed(2)
id.ao <- sample(which(ytrain==1),5)
id.aa <- sample(which(ytrain==0),5)
plot(xtrain[id.ao[1],],type="l",
     xlab="Frequency",ylab="Log-periodogram")
for(i in 2:5){
  lines(xtrain[id.ao[i],])
}
for(i in 1:5){
  lines(xtrain[id.aa[i],],col="red")
}
```



We run logistic regression and calculate the train and test errors.

```r
# logistic regression
fit <- glm(y~.,data=dtrain,family=binomial)
coef.glm <-  coefficients(fit)
pred_train <- as.numeric((predict(fit,type="response")>0.5))
pred_test <- as.numeric((predict(fit,type="response",newdata=dtest)>0.5))
mean(pred_train!=ytrain)
```

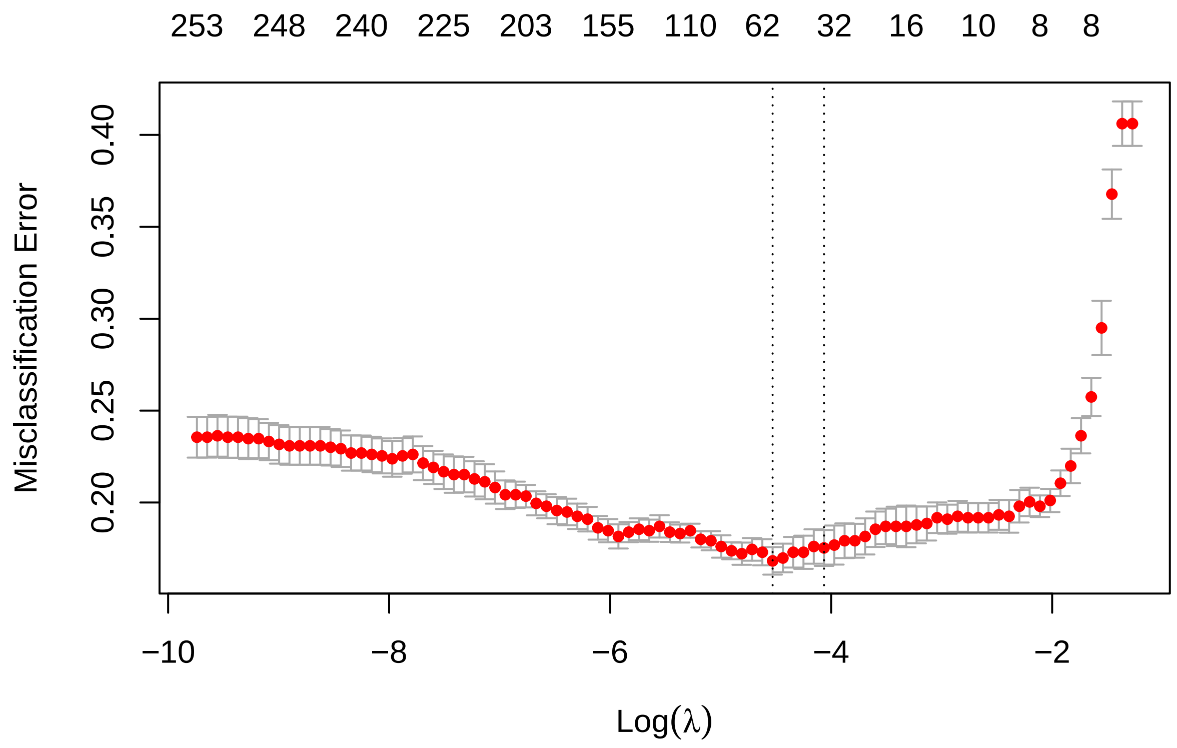
```
## [1] 0.09311424
```

```r
mean(pred_test!=ytest)
```

```
## [1] 0.2437358
```

We run Lasso regression and calculate the train and test errors.

```r
# lasso regression
fit.glmnet <-glmnet(xtrain,ytrain,family = "binomial",alpha=1)
cv.glmnet <- cv.glmnet(xtrain,ytrain,family = "binomial",type.measure = "class",
                       alpha = 1,nfolds = 10)
coef.lasso <- as.numeric(coefficients(fit.glmnet,s = cv.glmnet$lambda.1se))[-1]
plot(cv.glmnet)
```



```
pred_train <- c(predict(fit.glmnet,xtrain,s = cv.glmnet$lambda.1se,type = "class"))
pred_test <- c(predict(fit.glmnet,xtest,s = cv.glmnet$lambda.1se,type = "class"))
mean(pred_train!=ytrain)
```

```
## [1] 0.1564945
```

```
mean(pred_test!=ytest)
```

```
## [1] 0.1958998
```

We use the natural cubic spline basis with $\nu = 12$ to express the coefficients as a smooth function of the frequencies. We calculate the train and test errors.

```
# coefficient smoothing
hmat <- ns(x=1:256,df=12)
xstar <- xtrain%*%hmat
fit.smooth <- glm(dtrain$y~.,data=data.frame(xstar),family="binomial")
coef.smooth <- as.numeric(hmat%*%coef(fit.smooth)[-1])
pred_train <- as.numeric((predict(fit.smooth,type="response")>0.5))
pred_test <- as.numeric((predict(fit.smooth,type="response",
                                  newdata=data.frame(xtest%*%hmat))>0.5))

mean(pred_train!=ytrain)
```

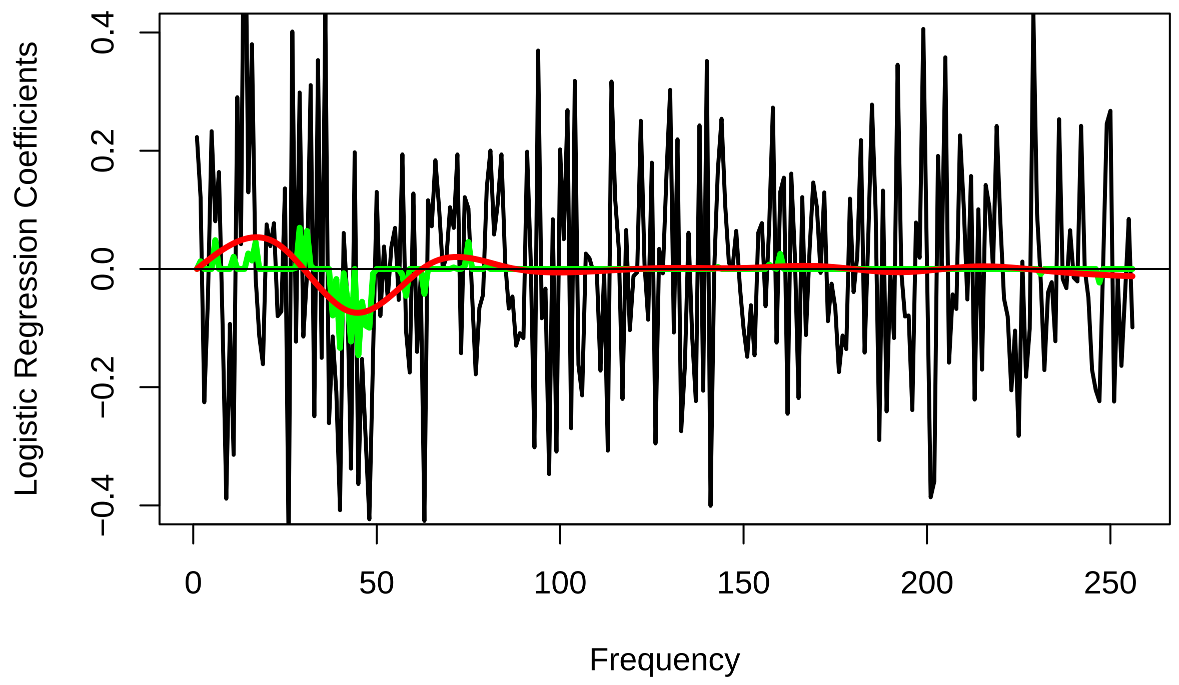
```
## [1] 0.1690141
```

```
mean(pred_test!=ytest)
```

```
## [1] 0.1867882
```

We plot the regression coefficients.

```
plot( coef.glm[-1],
      ylim=c(-0.4,+0.4),
      type="l",
      lwd=2,
      xlab="Frequency",
      ylab="Logistic Regression Coefficients" )
lines(coef.lasso,col="green",lwd=3)
lines(coef.smooth,col="red",lwd=3)
abline(h=0)
```



13 Classification and the `caret` package

Follow the short tutorial which guides you through a classification example using the `caret` package.

14 Survival analysis and the Lymphoma data

In this exercise we explore the Lymphoma data set to predict survival based on gene expression data.

1. Load the Lymphoma data and make a histogram of the survival times (`lymphx.txt` and `lymphtime.txt`) .
2. Plot the estimated survival curve using `survfit` (Kaplan-Meier method).
3. Fit a Cox regression model with the first three genes as predictors. Use the function `coxph`.
4. Build a predictive model using `glmnet` (data pre-processing: use the 100 genes with largest variance and standardize the resulting predictor matrix to have zero-mean and unit-variance). Which genes are selected? What is the C-index for the optimal tuning parameter?
5. Use the predictive model and classify patients into "good" and "poor" prognostic groups by thresholding the linear predictor at zero. Calculate the Kaplan-Meier curves for the two groups. What is your conclusion? Do you have any concerns?
6. The linear predictor scores computed in 5. are biased as they are evaluated on the same data for which they were computed. We now use a variant of cross-validation, known as *pre-validation*, in order to obtain a fair evaluation of the model. Calculate a pre-validated data set using the code below and calculate the Kaplan-Meier curves for patients with good and poor prognosis.

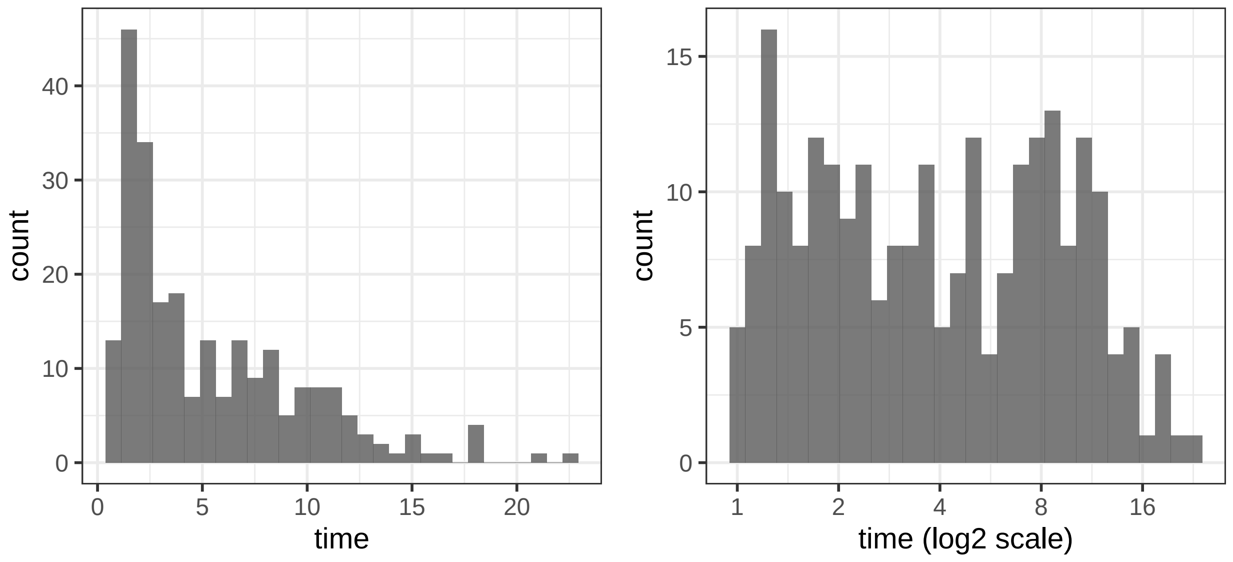
```r
# split data into K=5 folds
n.fold <- 5
foldid <- sample(rep(1:n.fold, length = nrow(x)))

# pre-validation
dat.preval <- data.frame(y)
dat.preval$lp <- NA

for (i in 1:n.fold){

  # train model on samples not in the kth fold
  omitk <- which(foldid==i)
  fitk <- cv.glmnet(x[-omitk,],y.surv[-omitk,],
                    family="cox",
                    type.measure="C",
                    nfolds = 5,
                    alpha=1)

  # calculated linear predictor on samples in the kth fold
  lp <- predict(fitk,
                newx=x[omitk,],
                s=fitk$lambda.1se,
                type="link")
  dat.preval$lp[omitk] <- lp
}
```

Solution to the exercise.

We load the data set.

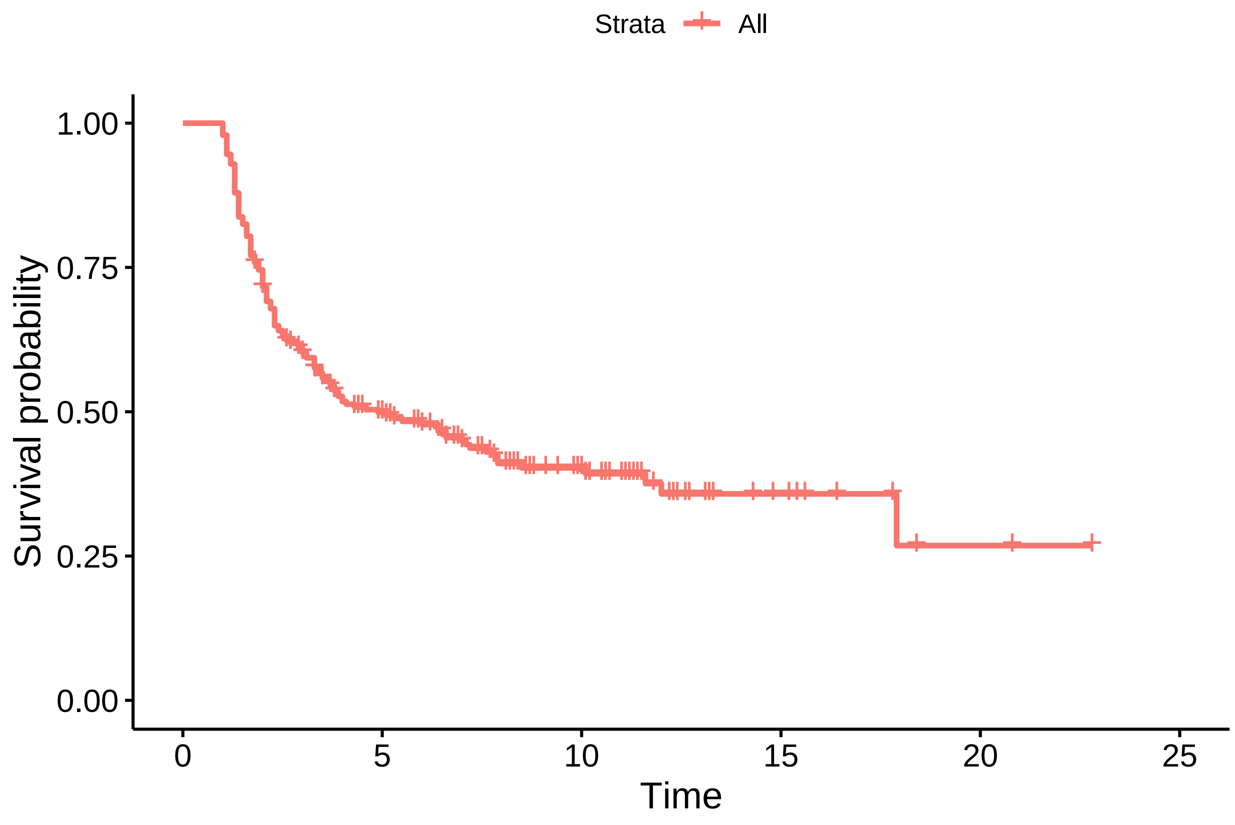
```r
# read gene expression matrix
x <- read.table("data/lymphx.txt")%>%
  as.matrix

# read survival data
y <- read.table("data/lymphtime.txt",header = TRUE)%>%
  as.matrix
```

Plot the distribution of the survival times.



Plot of the Kaplan-Meier estimates.

```r
library(survival) # survival analysis
library(survminer) # nice survival plots
dat <- data.frame(y)
fit.surv <- survfit(Surv(time, status) ~ 1,
                    data = dat)
ggsurvplot(fit.surv,conf.int=FALSE)
```



Fit a Cox regression model.

```
dat <- data.frame(cbind(y,x[,1:3]))
fit <- coxph(Surv(time,status)~.,data=dat)
summary(fit)
```

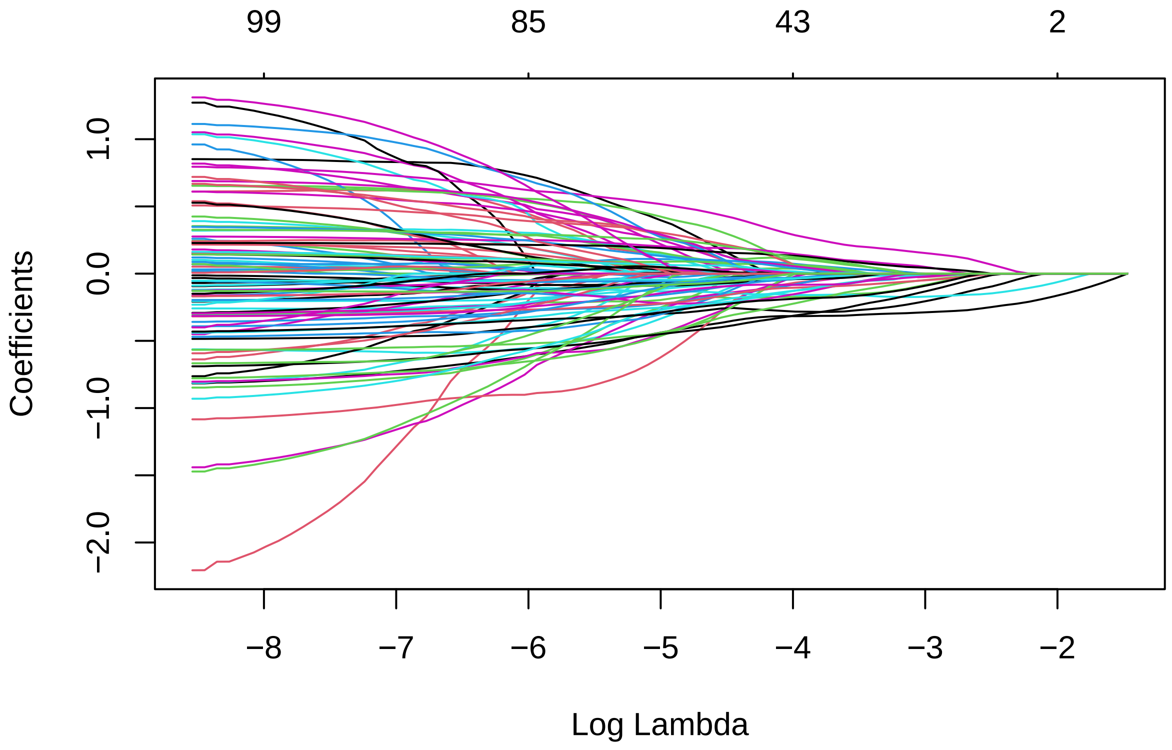
```
## Call:
## coxph(formula = Surv(time, status) ~ ., data = dat)
##
##   n= 240, number of events= 138
##
##       coef exp(coef) se(coef)      z Pr(>|z|)
## V1  0.6382    1.8931   0.4504  1.417    0.156
## V2 -0.5778    0.5611   0.4023 -1.436    0.151
## V3 -0.1508    0.8600   0.3785 -0.398    0.690
##
##    exp(coef) exp(-coef) lower .95 upper .95
## V1    1.8931     0.5282    0.7831     4.577
## V2    0.5611     1.7822    0.2551     1.234
## V3    0.8600     1.1627    0.4095     1.806
##
## Concordance= 0.559  (se = 0.028 )
## Likelihood ratio test= 4.46  on 3 df,   p=0.2
## Wald test            = 4.66  on 3 df,   p=0.2
## Score (logrank) test = 4.66  on 3 df,   p=0.2
```

Build a predictive model using `glmnet`. Data pre-processing.

```
# filter for top genes (highest variance) and scale the input matrix
topvar.genes <- order(apply(x,2,var),decreasing=TRUE)[1:100]
x <- scale(x[,topvar.genes])
```

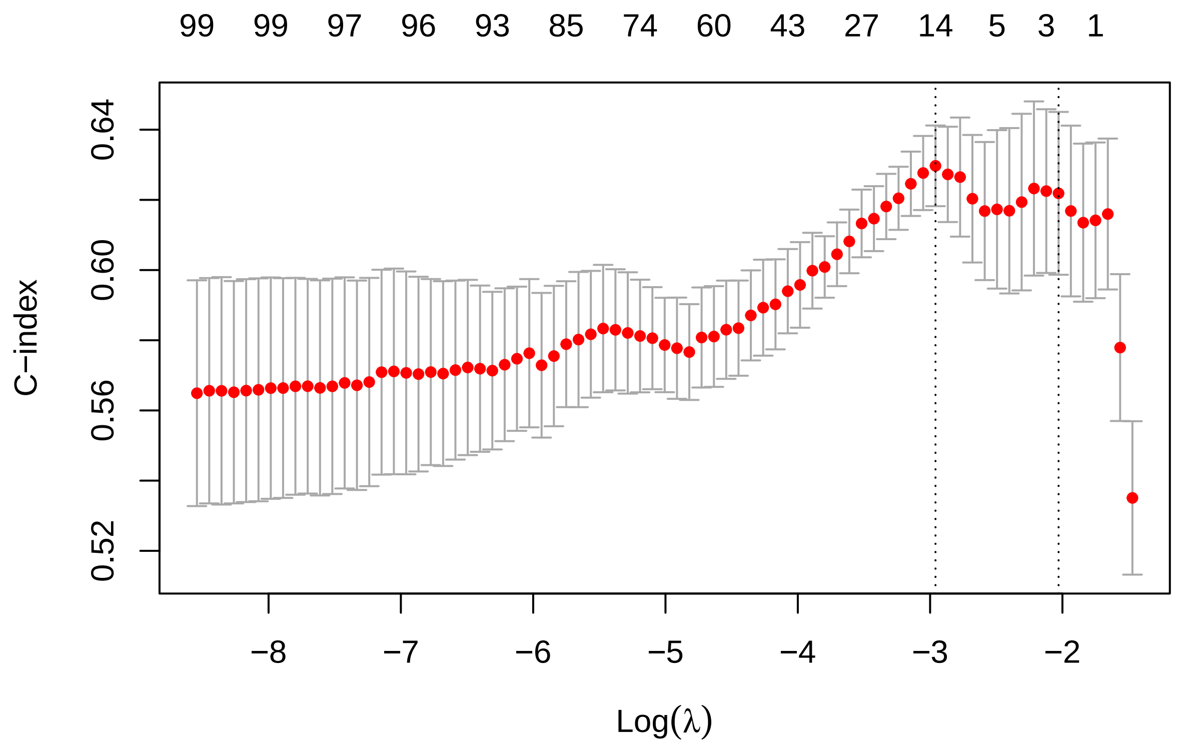
Run `glmnet` using `family="cox"`.

```
set.seed(1)
y.surv <- Surv(y[,"time"],y[,"status"])
fit.coxnet <- glmnet(x, y.surv, family = "cox")
plot(fit.coxnet,xvar="lambda")
```



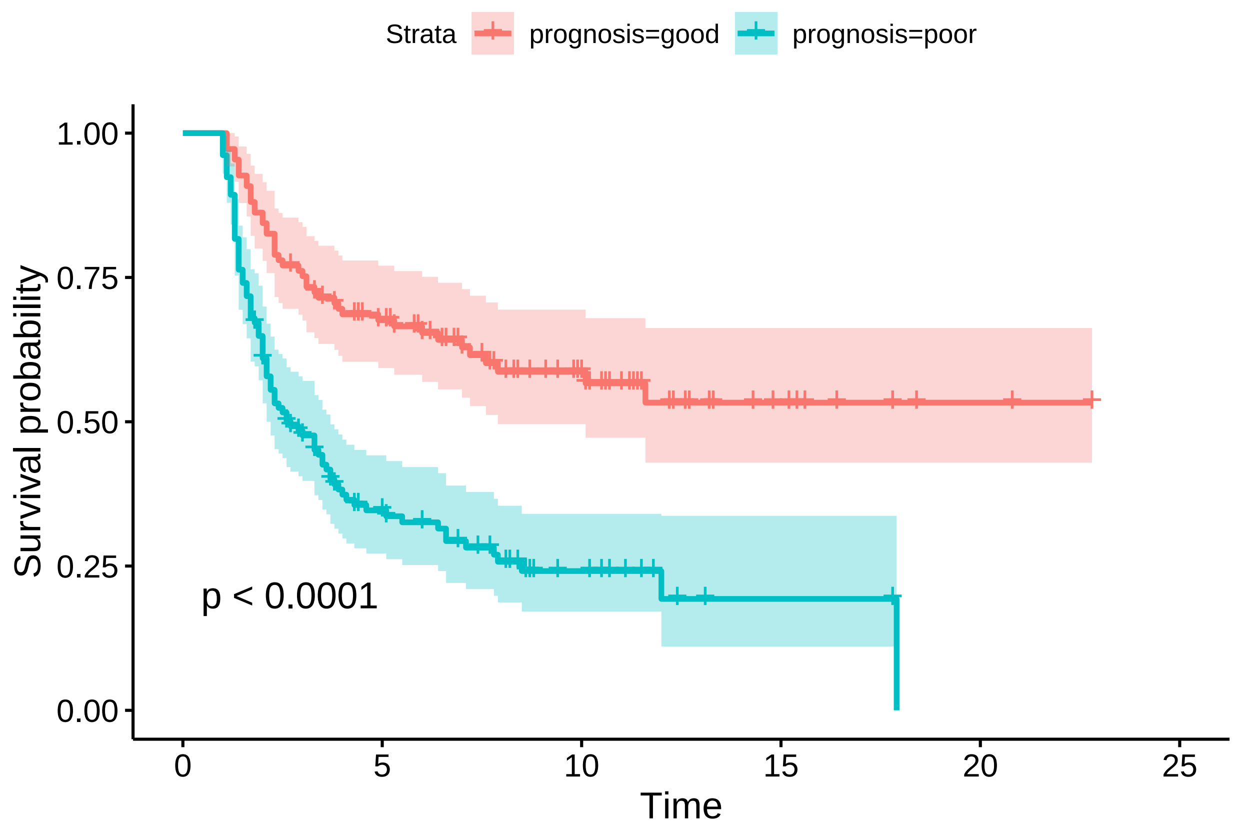
Calculate the cross-validated C-index.

```
cv.coxnet <- cv.glmnet(x,y.surv,
                       family="cox",
                       type.measure="C",
                       nfolds = 5)
plot(cv.coxnet)
```



Classify patients into groups with good and poor prognosis (based on thresholding the linear predictor at zero).

```
# linear predictor
lp <- predict(fit.coxnet,
              newx=x,
              s=cv.coxnet$lambda.1se,
              type="link")
dat <- data.frame(y)
dat$prognosis <- ifelse(lp>0,"poor","good")
fit.surv <- survfit(Surv(time, status) ~ prognosis,
                    data = dat)
ggsurvplot(fit.surv,conf.int = TRUE,pval=TRUE)
```



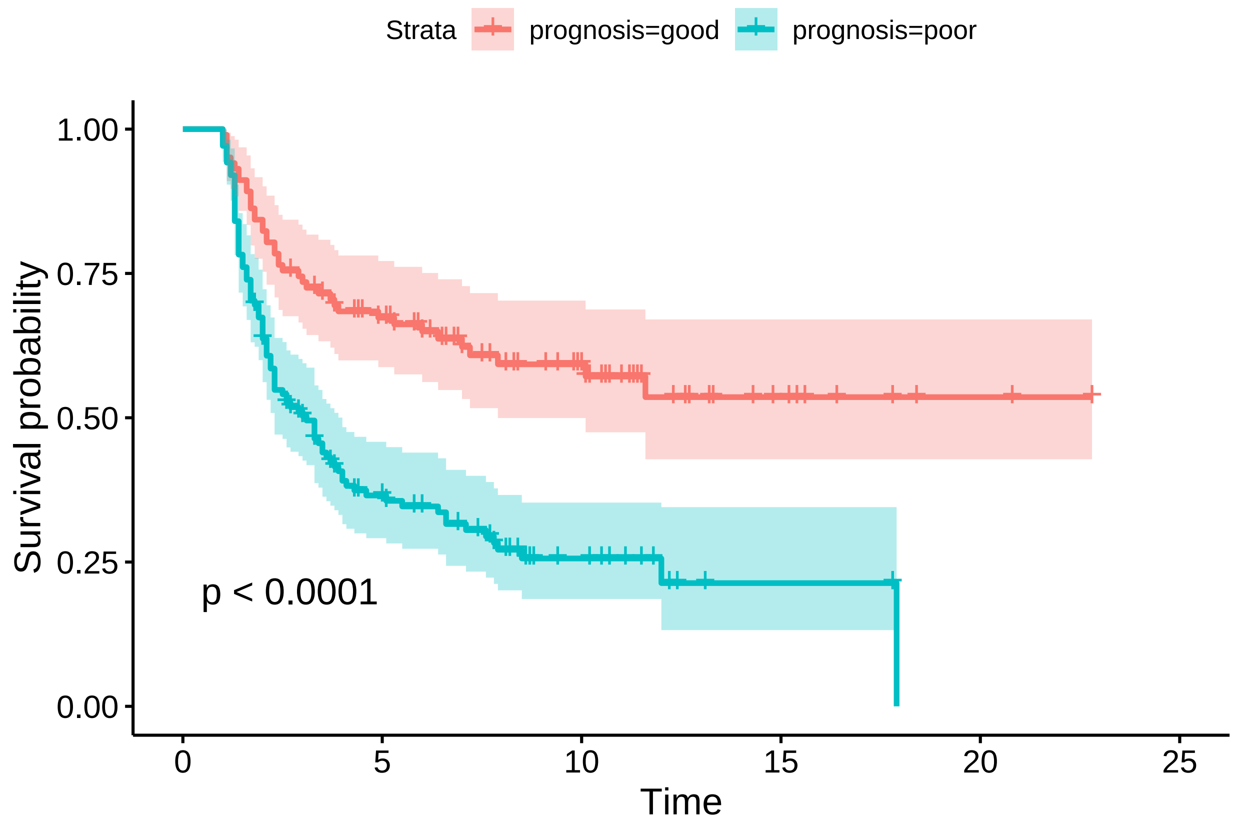
The curves are very well separated. However, these linear predictor scores are biased: we are evaluating their performance on the same data for which they were computed.

In order to obtain a fair evaluation of the model we calculate a pre-validated data set.

```
set.seed(150381)

# split data into K=5 folds
n.fold <- 5
foldid <- sample(rep(1:n.fold, length = nrow(x)))

# pre-validation
dat.preval <- data.frame(y)
dat.preval$lp <- NA

for (i in 1:n.fold){

  # train model on samples not in the kth fold
  omitk <- which(foldid==i)
  fitk <- cv.glmnet(x[-omitk,],y.surv[-omitk,],
                    family="cox",
                    type.measure="C",
                    nfolds = 5,
                    alpha=1)

  # calculated linear predictor on samples in the kth fold
  lp <- predict(fitk,
                newx=x[omitk,],
                s=fitk$lambda.1se,
                type="link")
  dat.preval$lp[omitk] <- lp
}
```

Plot the Kaplan-Meier curves for the good and poor prognostic groups based on the pre-validated data.

```
dat.preval$prognosis <- ifelse(dat.preval$lp>0,"poor","good")
fit.surv <- survfit(Surv(time, status) ~ prognosis,
                    data = dat.preval)
ggsurvplot(fit.surv,conf.int = TRUE,pval=TRUE)
```



15 Decision trees, Random Forest and AdaBoost

In this exercise we explore decision trees based on the South African heart disease data `sahd.rds`.

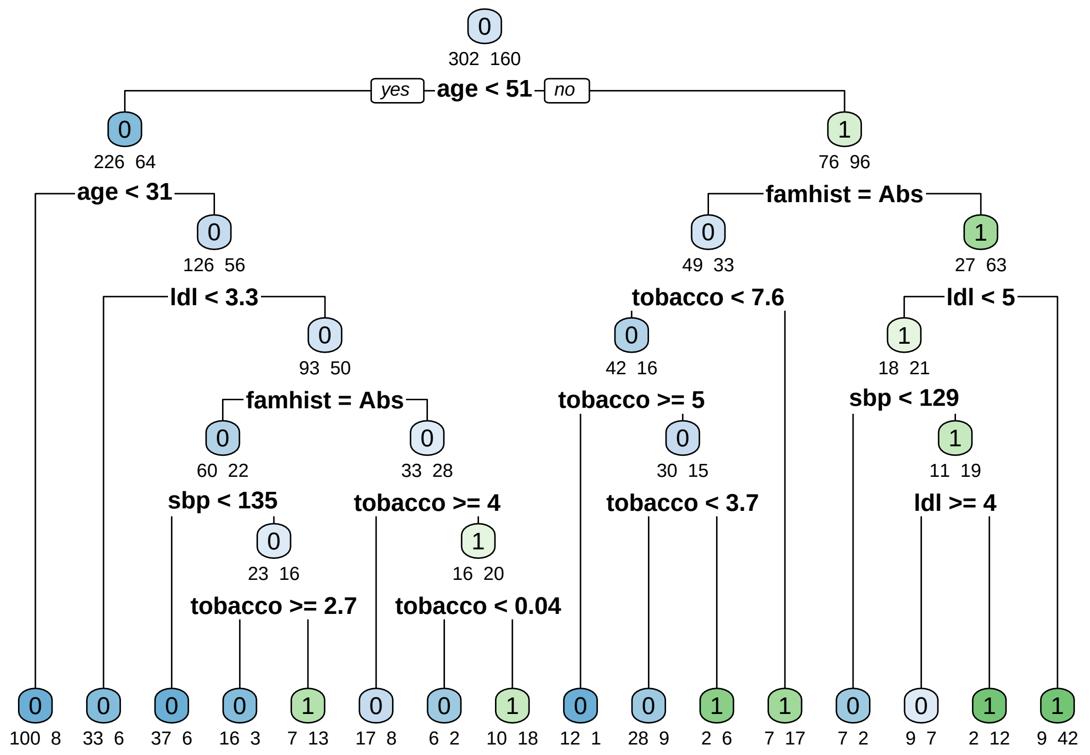
1. Load the South African heart disease data and grow a decision tree using `rpart`. Visualize the tree using `rpart.plot`. How many leave nodes has the tree?
2. Re-grow the tree but now relax the "default" control parameters (choose `rpart.control(cp=0, minsplit=50)`). How many leaves has the tree now?

3. Plot the cross-validation error against the complexity parameter α. What is the tree size of the optimal model?
4. Prune the tree using `prune` and by choosing cp ($=\alpha$) to achieve minimal cross-validation error.
5. Calculate the confusion matrix and the misclassification error.
6. Generate a bootstrap sample. Grow a tree and calculate the out-of-bag error.
7. Fit a random forest using `randomForest`. Plot the fitted object. What is this plot telling us? Calculate the variable importance. Which are the most important variables?
8. Run AdaBoost using `gbm`. What is the prediction for a patient with covariates sbp=100, tobacco=0, ldl=5, famhist="Present", obesity=25, alcohol=10 and age=50. Compute the variable importance.

The solution to this exercise.

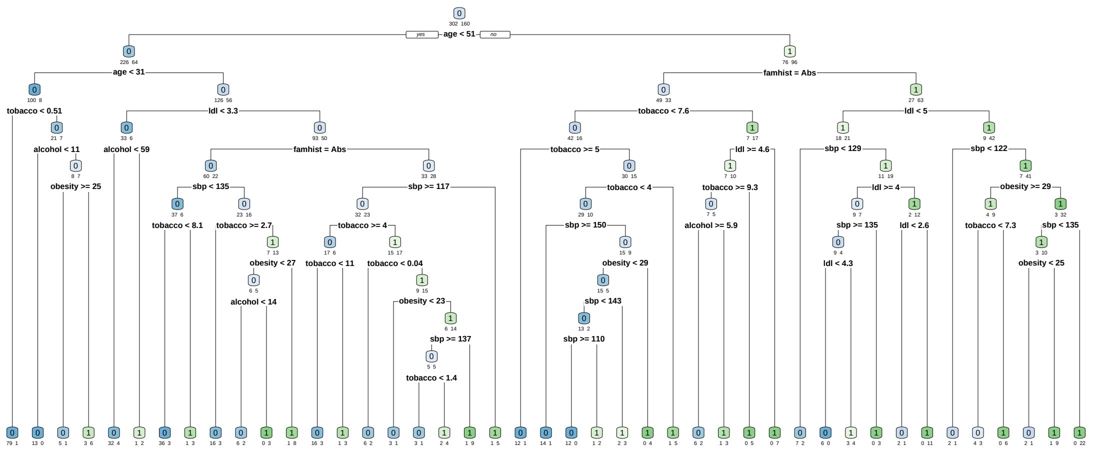
First we read the data and grow the tree.

```r
# load packages
library(rpart)
library(rpart.plot)

# read data set
sahd <- readRDS(file="data/sahd.rds")

# grow a classification tree
fit.tree <- rpart(chd~.,data=sahd,method="class")
rpart.plot(fit.tree,extra=1,under=TRUE,tweak = 1.2,faclen=3)
```



We re-grow the tree using different control parameters.

```r
# controlling the growth of the tree with rpart.control
# cp: improvement in each split needs to be > cp
# minsplit: minimal number of samples in a node in order to do a split
fit.tree2 <- rpart(chd~.,data=sahd,method="class",
                   control = rpart.control(cp = 0,minsplit=10)

)
rpart.plot(fit.tree2,extra=1,under=TRUE,tweak = 1.2,faclen=3)
```



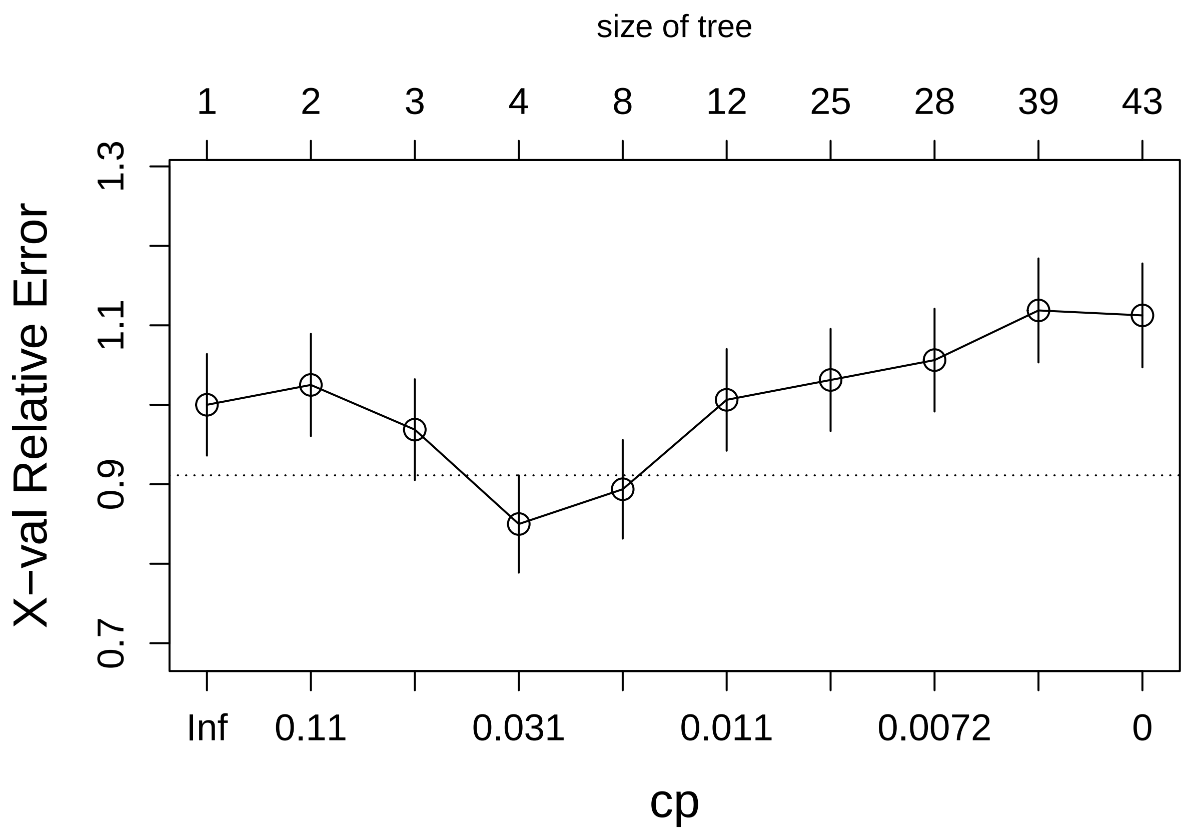
We can get the tree size from the cptable (tree size=number of leaves=number splits+1).

```r
fit.tree2$cptable[fit.tree2$cptable[,"CP"]==0,"nsplit"]+1 # number of leaves
```

```
## [1] 43
```

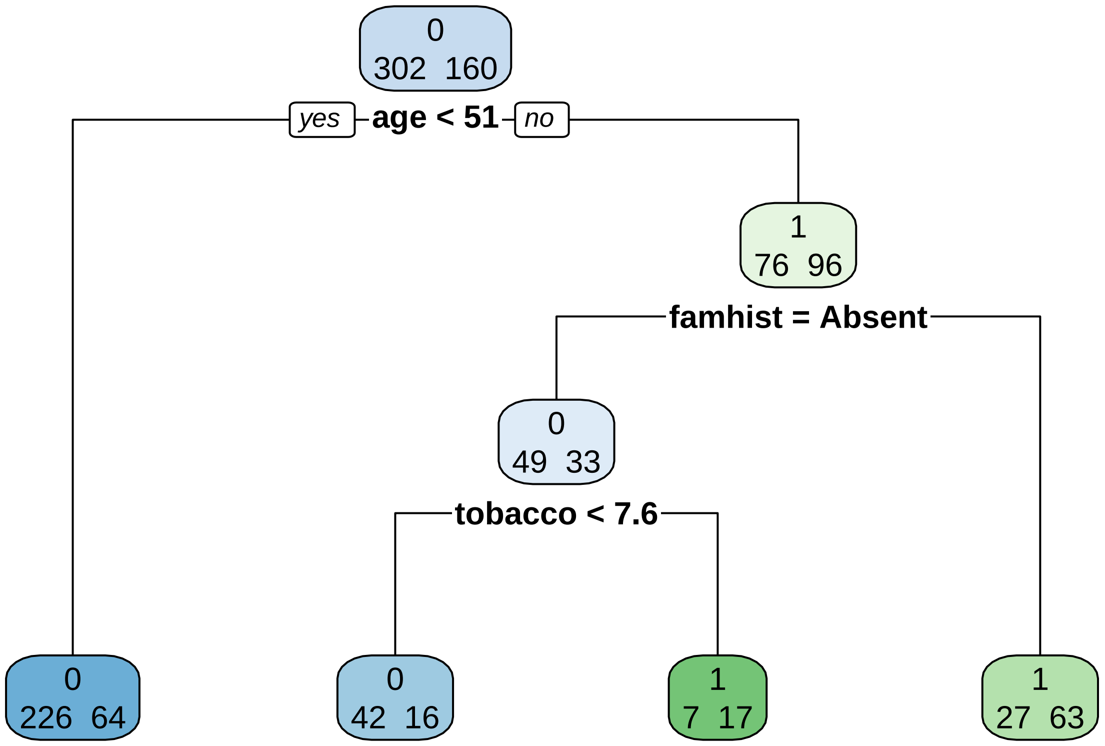
Next, we plot the cross-validation error against the complexity parameter α. A tree of size 4 has smallest cross-validation error.

```r
plotcp(fit.tree2,cex.lab=1.5,cex.axis=1.2,cex=1.5)
```



We prune the tree and visualize the result.

```
# prune the tree
fit.prune<- prune(fit.tree2,
                  cp=fit.tree2$cptable[which.min(fit.tree2$cptable[,"xerror"]),"CP"])
rpart.plot(fit.prune,extra=1)
```



Finally, we compute the confusion matrix and the misclassification error.

```
# confusion matrix of actual and fitted class labels
table(Actual=sahd$chd,Fitted=predict(fit.prune,type="class"))
```

```
##        Fitted
## Actual   0   1
##       0 268  34
##       1  80  80
```

```
# misclassification error
mean(sahd$chd!=predict(fit.prune,type="class"))
```

```
## [1] 0.2467532
```

We sample with replacement (bootstrap sample).

```
set.seed(1)
inthebag <- sample(1:nrow(sahd),size=nrow(sahd),replace=TRUE)
outofbag <- setdiff(1:nrow(sahd),inthebag)
```

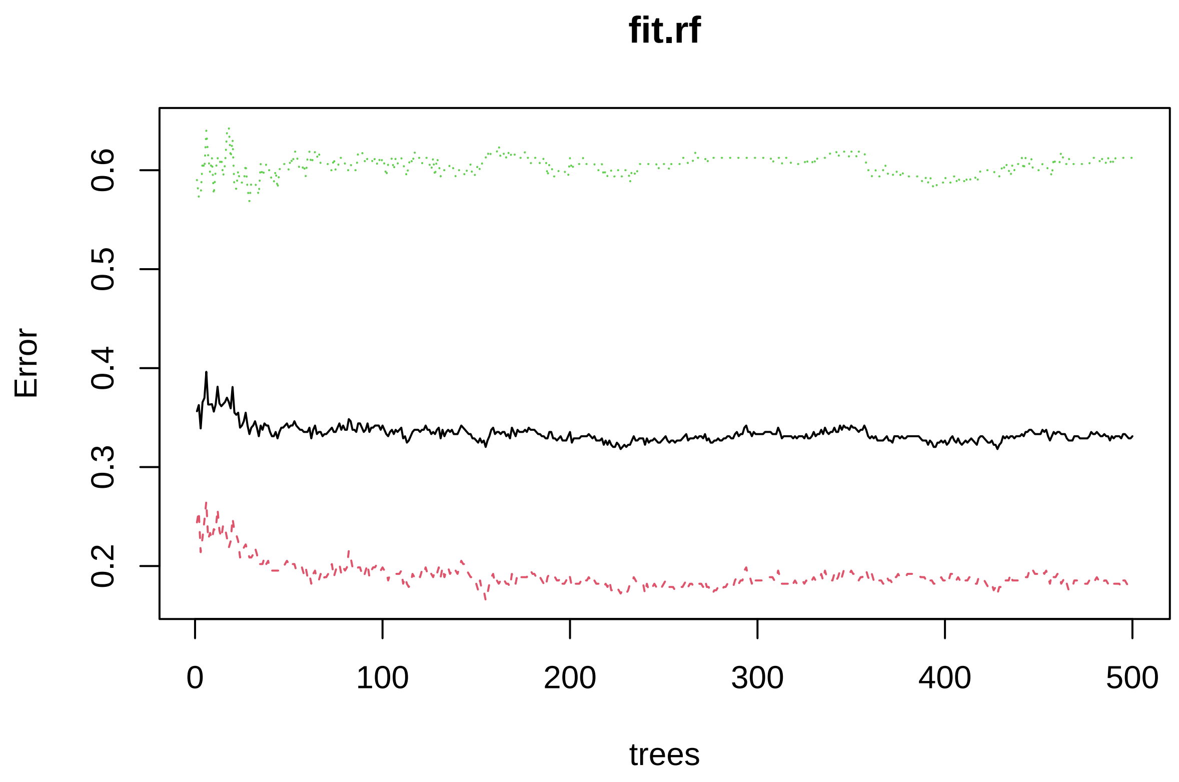
We fit a tree on the in-the-bag samples and calculate the misclassification error on the out-of-bag samples.

```r
fit.in <- rpart(chd~.,data=sahd[inthebag,],method="class")
pred.oob <- predict(fit.in,newdata=sahd[outofbag,],type="class")
mean(sahd$chd[outofbag]!=pred.oob)
```

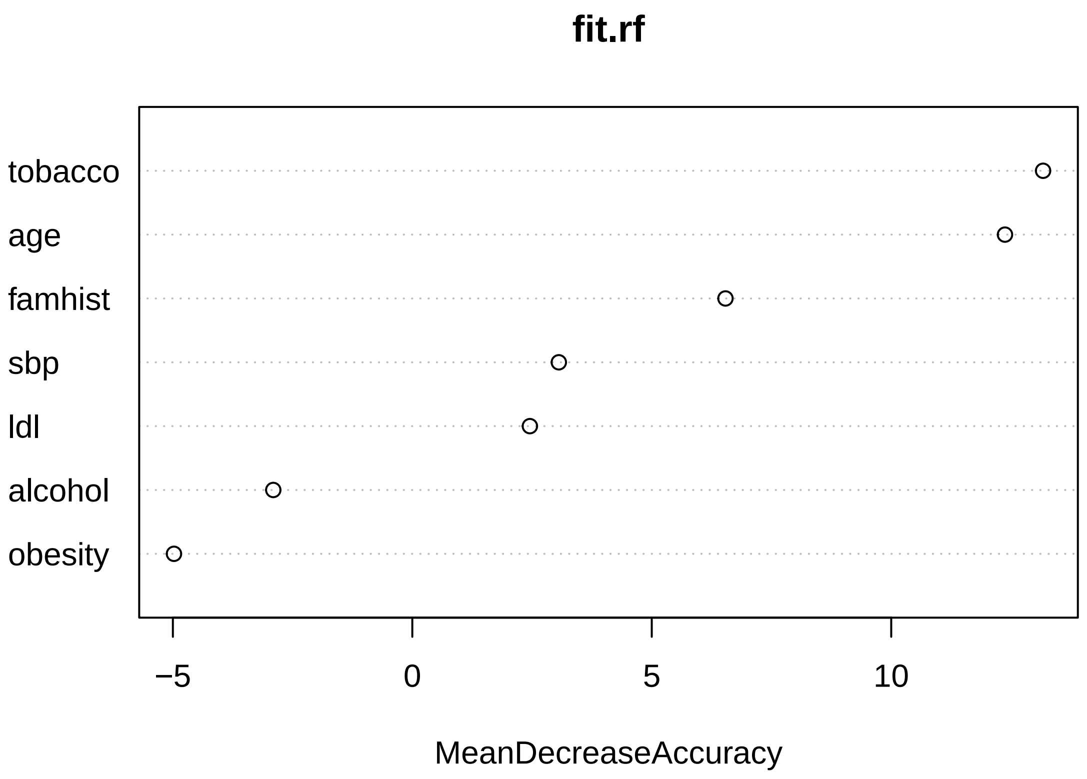
```
## [1] 0.3559322
```

We fit the random forest, plot the error as a function of the number of trees and plot the variable importance. The out-of-bag error estimate (black line) stabilizes with increasing number of fitted trees.

```r
library(randomForest)
sahd$chd <- factor(sahd$chd)
fit.rf <- randomForest(chd~.,data=sahd,importance=TRUE)
plot(fit.rf)
```

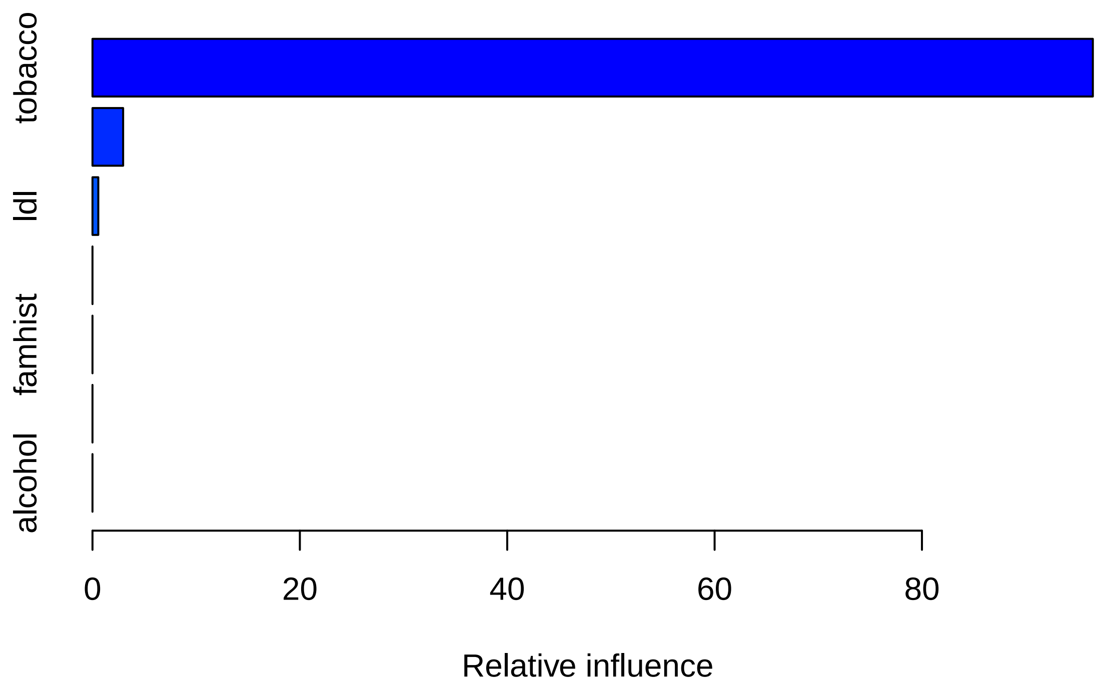


```r
varImpPlot(fit.rf,type=1)
```



We run AdaBoost using `gbm` and by specifying `distribution = "adaboost"`. The `summary` provides a measure of variable importance. Prediction can be made using `predict`.

```
library(gbm)
fit.boost <-gbm(chd~.,
                data=sahd,
                distribution = "adaboost")# note: for adaboost the outcome must be numeric
summary(fit.boost)
```



```
##              var    rel.inf
## tobacco tobacco 96.4821033
## age         age  2.9567803
## ldl         ldl  0.5611164
## sbp         sbp  0.0000000
## famhist famhist  0.0000000
## obesity obesity  0.0000000
## alcohol alcohol  0.0000000
```

```
newd <- data.frame(sbp=100,tobacco=0,ldl=5,famhist=factor("Present"),obesity=25,alcohol=10,age=50)
predict(fit.boost,
        newdata=newd,
        type="response" )
```

```
## Using 100 trees...
```

```
## [1] 1
```

```
fit.boost <-gbm(chd~.,
                data=sahd,
                distribution = "bernoulli",# note: for adaboost the outcome must be numeric
                 #n.trees=1000,
                 #interaction.depth=1,
                 #shrinkage=0.1,
                 #cv.folds=5,# iteraction.depth=1 <-> stump
                ) # number of boosting iterations
summary(fit.boost)
newd <- data.frame(sbp=100,tobacco=0,ldl=5,famhist=factor("Present"),obesity=25,alcohol=10,age=50)
predict(fit.boost,
        newdata=newd,
        type="response" )

# find index for n trees with minimum CV error
min_error <- which.min(fit.boost$cv.error)

# get MSE and compute RMSE
fit.boost$cv.error[min_error]
## [1] 23112.1

# plot loss function as a result of n trees added to the ensemble
gbm.perf(fit.boost, method = "cv")
```

16 Email spam and data mining

In this exercise we explore a typical data mining task. We use the `spam.rds` data set. The data for this example consists of information from 4601 email messages. The aim is to predict whether the email is spam. For all 4601 email messages, the true outcome (email type) “email” or “spam” is available, along with the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message. This is a supervised learning problem, with the outcome the class variable email/spam. It is also called a classification problem.

1. Read the data set and partition it into 2/3 of training and 1/3 of test data.
2. Use `rpart` to fit a decision tree. Examine the cross-validation output, prune the tree and calculate the misclassification error.
3. Use `randomForest`, calculate the misclassification error and plot the variable importance.
4. Run AdaBoost using `gbm`, print the misclassication error and plot the relative influence of the variables.

The solution to the exercise.

We read the data and divide into 2/3 training and 1/3 test data.

```
spam <- readRDS(file="data/spam.rds")
set.seed(102)
train <- sort(sample(nrow(spam),3065))
spam.train<- spam[train,]
spam.test<- spam[-train,]
```

We grow a classification tree and examine the 10-fold cross-validation output.

```
# grow tree
library(rpart)
fit.rpart <- rpart(spam~.,data=spam.train,method='class',cp=1e-5)

# cross-validation output
t.tab <- printcp(fit.rpart)
```

```
##
## Classification tree:
## rpart(formula = spam ~ ., data = spam.train, method = "class",
##     cp = 1e-05)
##
## Variables actually used in tree construction:
##  [1] capavg      caplong     captot      cf.dollar   cf.exclaim  wf.000      wf.650      wf.all
## [10] wf.edu      wf.free     wf.george   wf.hp       wf.hpl      wf.internet wf.money    wf.our
## [19] wf.pm       wf.re       wf.remove   wf.you      wf.your
##
## Root node error: 1223/3065 = 0.39902
##
## n= 3065
##
##            CP nsplit rel error  xerror     xstd
## 1  0.48569092      0   1.00000 1.00000 0.022167
## 2  0.06868357      1   0.51431 0.53557 0.018556
## 3  0.06132461      2   0.44563 0.44481 0.017296
## 4  0.02698283      4   0.32298 0.33279 0.015361
## 5  0.02371218      5   0.29599 0.31562 0.015019
## 6  0.01635323      6   0.27228 0.29027 0.014486
## 7  0.01471791      7   0.25593 0.28618 0.014397
## 8  0.00735895      8   0.24121 0.26574 0.013937
## 9  0.00572363      9   0.23385 0.26165 0.013842
## 10 0.00490597     11   0.22240 0.25675 0.013727
## 11 0.00408831     13   0.21259 0.24612 0.013471
## 12 0.00327065     19   0.18806 0.24285 0.013391
## 13 0.00245298     22   0.17825 0.23385 0.013167
## 14 0.00218043     24   0.17334 0.22567 0.012958
## 15 0.00163532     27   0.16680 0.22077 0.012830
## 16 0.00130826     36   0.15127 0.21668 0.012722
## 17 0.00081766     41   0.14473 0.21504 0.012679
## 18 0.00040883     47   0.13982 0.20932 0.012524
## 19 0.00001000     51   0.13818 0.20769 0.012480
```

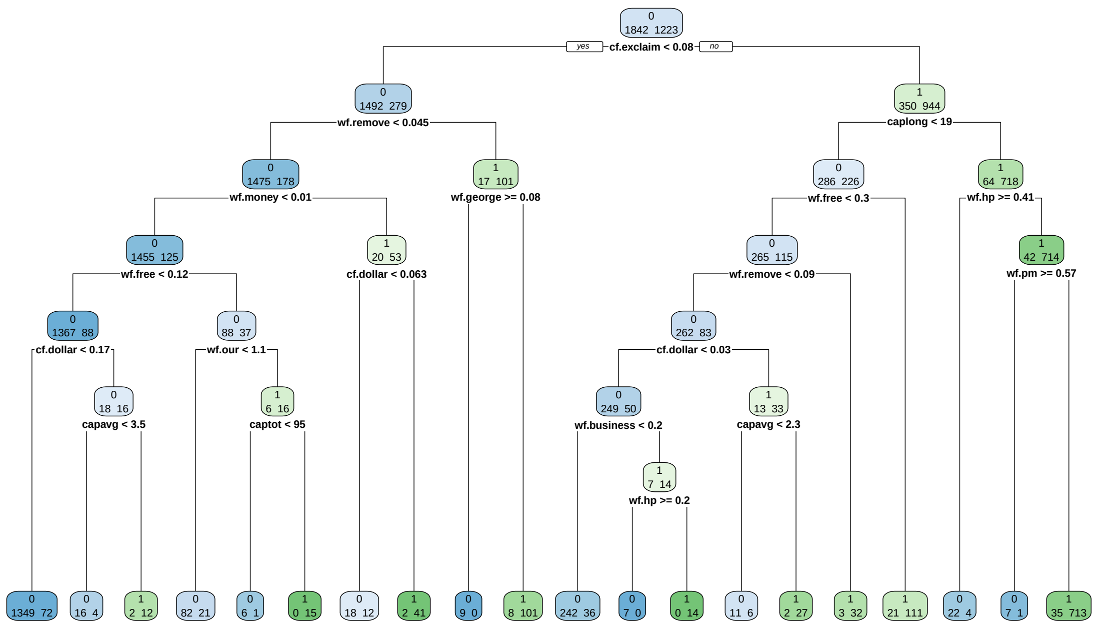
```
kable(t.tab,digits = 3,booktabs=TRUE)
```

CP	nsplit	rel error	xerror	xstd
0.486	0	1.000	1.000	0.022
0.069	1	0.514	0.536	0.019
0.061	2	0.446	0.445	0.017
0.027	4	0.323	0.333	0.015
0.024	5	0.296	0.316	0.015
0.016	6	0.272	0.290	0.014
0.015	7	0.256	0.286	0.014
0.007	8	0.241	0.266	0.014
0.006	9	0.234	0.262	0.014
0.005	11	0.222	0.257	0.014
0.004	13	0.213	0.246	0.013
0.003	19	0.188	0.243	0.013
0.002	22	0.178	0.234	0.013
0.002	24	0.173	0.226	0.013

CP	nsplit	rel error	xerror	xstd
0.002	27	0.167	0.221	0.013
0.001	36	0.151	0.217	0.013
0.001	41	0.145	0.215	0.013
0.000	47	0.140	0.209	0.013
0.000	51	0.138	0.208	0.012

We prune the tree and visualize the result.

```r
fit.rpart2 <- prune(fit.rpart,cp=.0033)
rpart.plot(fit.rpart2,extra=1)
```



We compute the misclassification error.

```r
(err.rpart2 <- mean(spam.test$spam!=predict(fit.rpart2,newdata=spam.test,type="class")))
```

```
## [1] 0.09570312
```

We fit the random forest, calculate the misclassification error and plot the variable importance.

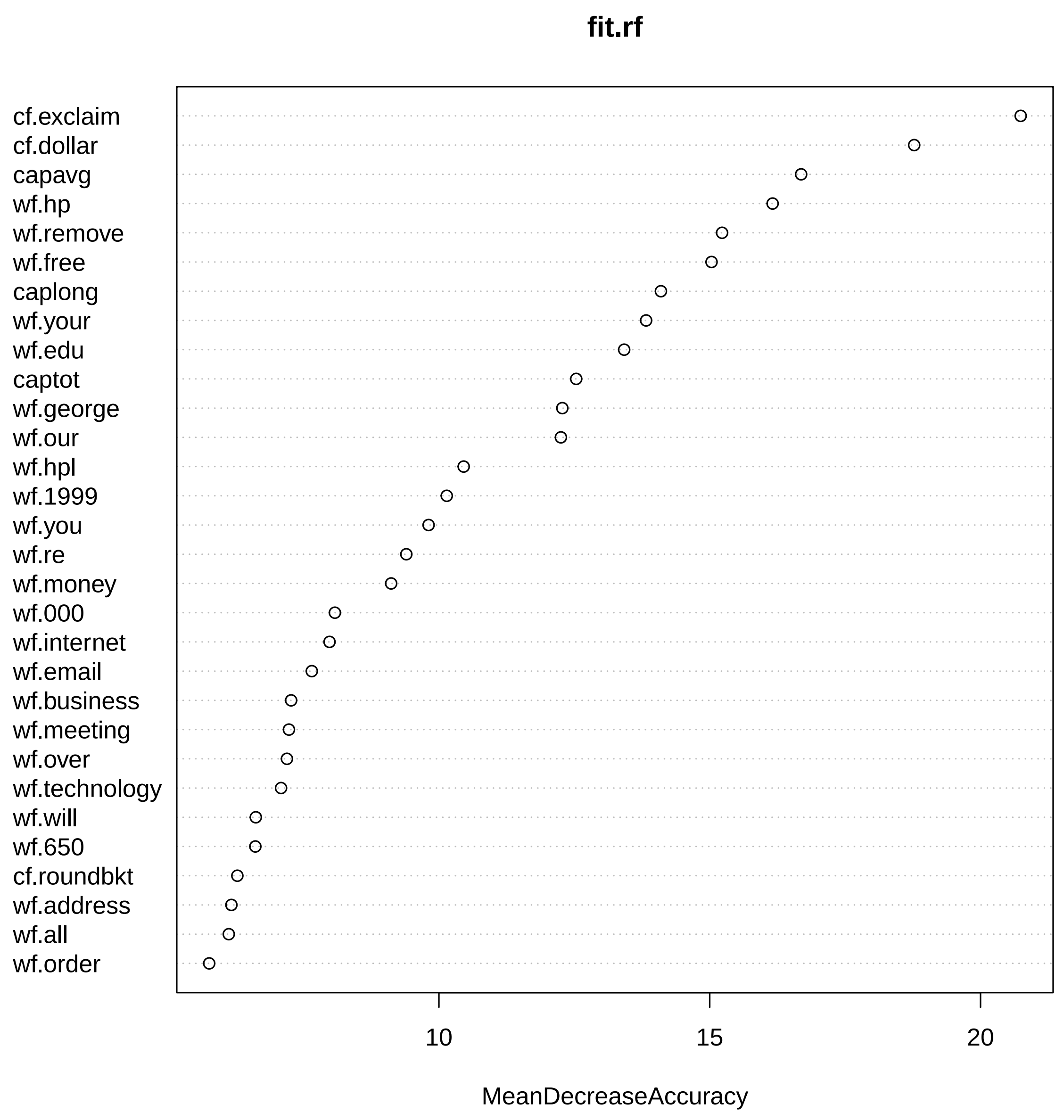
```
fit.rf <- randomForest(spam ~ .,
                       data = spam.train%>%
                         dplyr::mutate(spam=factor(spam)),
                       ntree = 100,
                       importance=TRUE)
fit.rf
```

```
##
## Call:
##  randomForest(formula = spam ~ ., data = spam.train %>% dplyr::mutate(spam = factor(spam)),      ntre
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 4.96%
## Confusion matrix:
##      0    1 class.error
## 0 1787   55  0.02985885
## 1   97 1126  0.07931316
```

```
(err.rf <- mean(spam.test$spam!=predict(fit.rf,newdata=spam.test)))
```

```
## [1] 0.05533854
```

```
varImpPlot(fit.rf,type=1)
```



We run gradient boosting `gbm` with `distribution = "bernoulli"` and tuning parameters `shrinkage=0.1`, `interaction.depth=3` and `n.trees=1000`. In addition we perform cross-validation `cv.folds=5` in order to monitor performance as a function of the boosting iterations.

```
fit.boost <-gbm(spam~.,
                data=spam.train,
                distribution = "bernoulli",
                shrinkage=0.1,
                n.trees=1000,
                interaction.depth = 3,
                cv.folds=5)
fit.boost
```

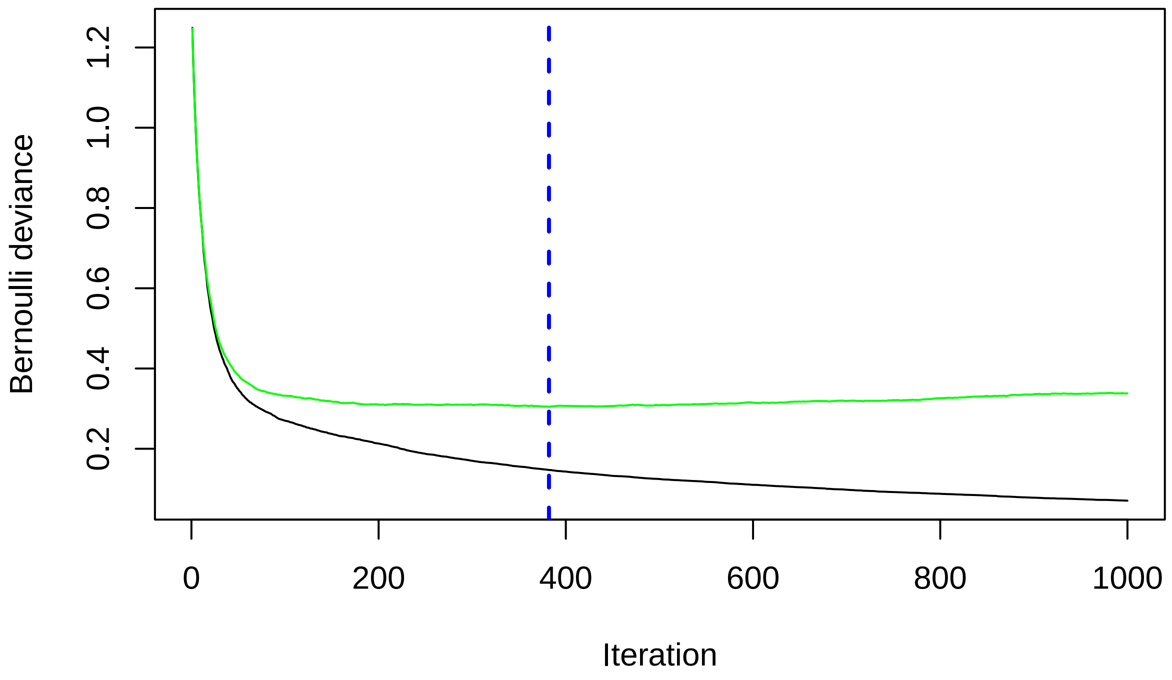
```
## gbm(formula = spam ~ ., distribution = "bernoulli", data = spam.train,
##     n.trees = 1000, interaction.depth = 3, shrinkage = 0.1, cv.folds = 5)
## A gradient boosted model with bernoulli loss function.
## 1000 iterations were performed.
## The best cross-validation iteration was 382.
## There were 57 predictors of which 50 had non-zero influence.
```

We explore the performance as a function of the number of boosting iterations.

```
# find index for n trees with minimum CV error
which.min(fit.boost$cv.error)
```

```
## [1] 382
```

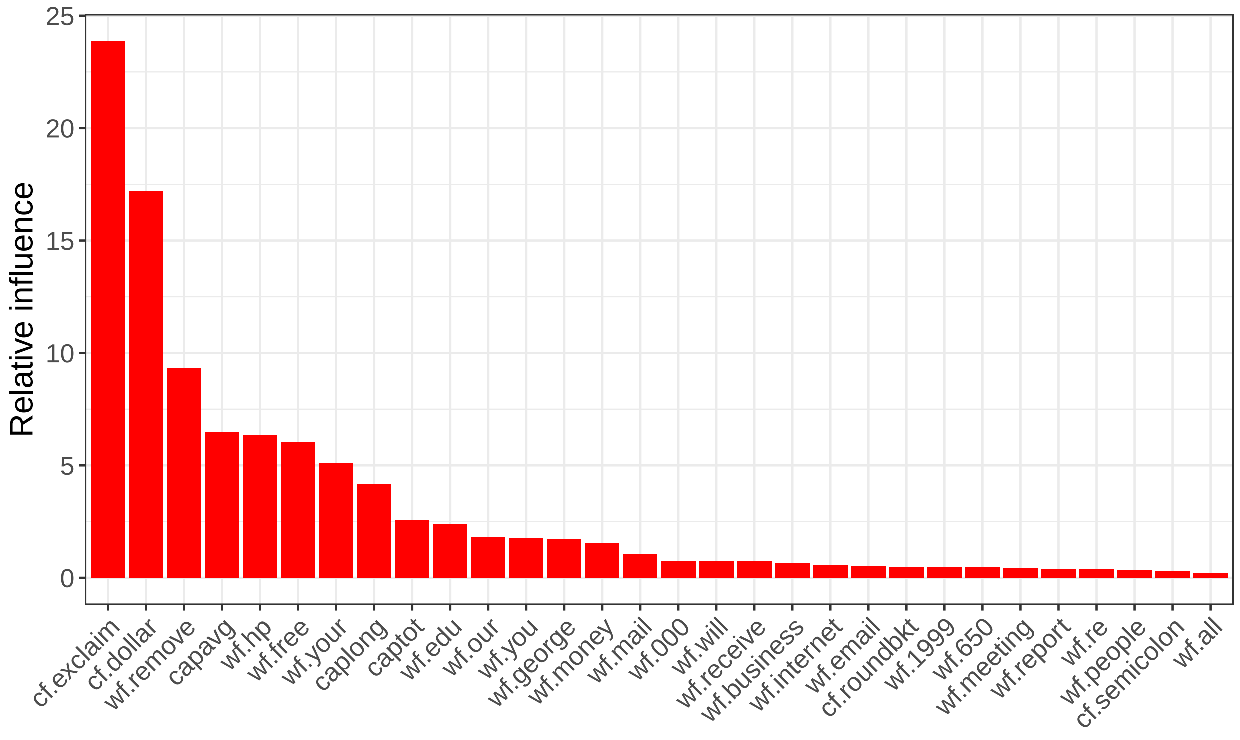
```
# plot loss as a function of the number of boosting iteration
gbm.perf(fit.boost, method = "cv")
```



```
## [1] 382
```

We plot the relative influence of each variable:

```
drelimp <- summary(fit.boost,plotit=FALSE)%>%
  data.frame

drelimp2 <- drelimp%>%
  slice(1:30)

drelimp2%>%
  ggplot(.,aes(x=reorder(var,-rel.inf),y=rel.inf))+
  geom_bar(stat="identity",fill="red")+
  theme_bw()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        text = element_text(size=15))+
  ylab("Relative influence")+
  xlab("")
```



Finally, we calculate the misclassification error.

```
pred.boost <- ifelse(predict(fit.boost, newdata = spam.test,type="response")>0.5,1,0)
```

```
## Using 382 trees...
```

```
(err.adaboost <- mean(spam.test$spam!=pred.boost))
```

```
## [1] 0.05143229
```

17 Multiple testing and gene expression

In this exercise we explore the issue of multiple testing using the gene expression data from the mouse experiment (see lecture slides).

1. Load the gene expression data set `esetmouse.rds` and explore the structure of the object (use the functions `str`, `dim`, `pData`, `expr`).
2. Carry out a two-sample t-test for a single gene and print the p-value.
3. Repeat 2. for all genes, generate a histogram of p-values and calculated the number of p-values < 0.05.
4. Perform multiple testing correction using the Bonferroni and FDR methods and count the number of significant genes (`p.adjust`).
5. Use the `limma` package to run the differential expression analysis and plot the result using a volcano plot (use the functions `lmFit`, `eBayes` and `volcanoplot`)

Solution to the exercise.

We load the `ExpressionSet`.

```
library(Biobase)
esetmouse <- readRDS(file="data/esetmouse.rds")
class(esetmouse)
```

```
## [1] "ExpressionSet"
## attr(,"package")
## [1] "Biobase"
```

```
dim(esetmouse)
```

```
## Features  Samples
##    15923       24
```

We can look at the expression values of the first sample and the first 6 genes.

```
exprs(esetmouse)[1:6,1]
```

```
## 1367452_at 1367453_at 1367454_at 1367455_at 1367456_at 1367457_at
##  10.051651  10.163334  10.211724  10.334899  10.889349   9.666755
```

An overview on the phenotype data can be obtained using the following commands.

```
table(pData(esetmouse)$strain)
```

```
##
##  A  B
## 12 12
```

We run a two-sample t-test for gene $j = 11425$.

```
x <- esetmouse$strain # strain information
y <- t(exprs(esetmouse)) # gene expressions matrix (columns refer to genes)

ttest <- t.test(y[,11425]~x,var.equal=TRUE)
ttest$statistic #tscore
```

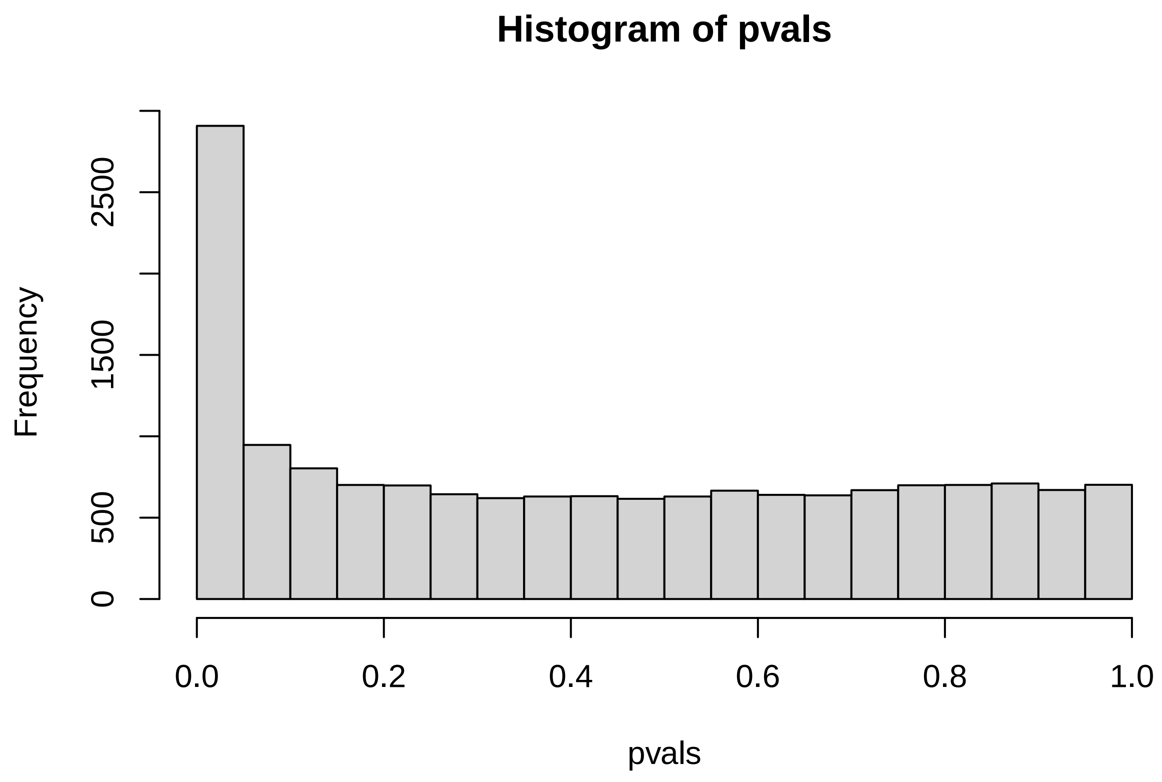
```
##        t
## 1.774198
```

```
ttest$p.value
```

```
## [1] 0.0898726
```

We calculate p-values for all genes, plot them as a histogram and count the number < 0.05.

```
pvals <- apply(y,2,FUN=
                 function(y){
                   t.test(y~x,var.equal=TRUE)$p.value
                 })
hist(pvals)
```



```
sum(pvals<0.05)
```

```
## [1] 2908
```

We run the Bonferroni and FDR correction methods.

```
sum(p.adjust(pvals,method="bonferroni")<0.05)
```

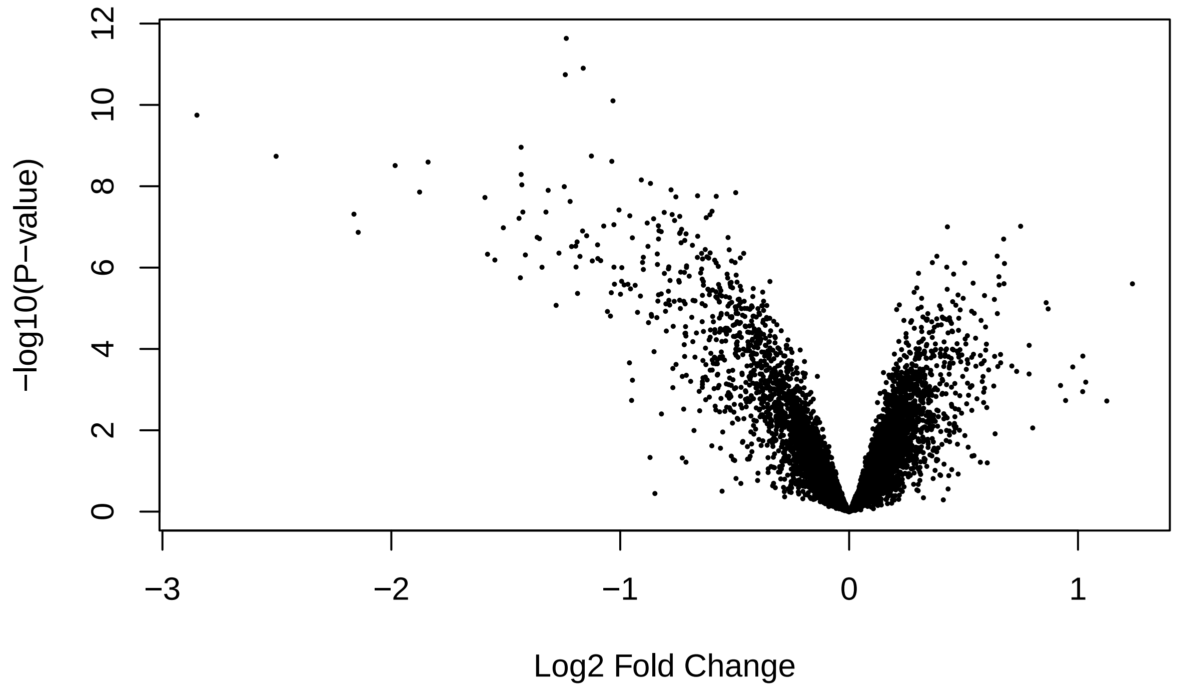
```
## [1] 82
```

```r
sum(p.adjust(pvals,method="fdr")<0.05)
```

```
## [1] 1123
```

Finally, we use the `limma` package to perform the differential expression analysis and we plot the results using a volcano plot.

```r
library(limma)
# first argument: gene expression matrix with genes in rows and sample in columns
# second argument: design matrix
fit.limma <- lmFit(t(y), design=model.matrix(~ x))

# eBayes
ebfit <- eBayes(fit.limma)

# volcanplot
volcanoplot(ebfit,coef=2)
```


−log10(P−value)
Log2 Fold Change