proof 3

Marcell Frisch

2023 10 16

Task 1

We start by focusing on the ridge expression. In the first part we want to center the response and the regressors:

$$\sum_{i=1}^{N}\left(y_i-\beta_0-\sum_{j=1}^{p}(x_{ij}-\overline{x}_j)\beta_j\right)=$$
$$\sum_{i=1}^{N}\left(y_i-\overline{y}-\beta_0-\sum_{j=1}^{p}\overline{x_j}\beta_j-\sum_{j=1}^{p}(x_{ij}-\overline{x}_j)\beta_j\right)=$$
$$\sum_{i=1}^{N}\left(y_i-\beta_0^c-\sum_{j=1}^{p}(x_{ij}-\overline{x}_j)\beta_j^c\right).$$

Hence, we have that:

$$\beta_0^c=\beta_0+\sum_{j=1}^{p}\overline{x_j}\beta_j-\overline{y}$$
$$\beta_j^c=\beta_j\ \ j=1,2,\ldots,p$$

As a final remark, this centering procedure consist in shifting all the variables x_j and the response y to have mean zero. As a result, only β_0^c which is the intercept, is going to change to be equal to 0, whereas the slope of the regression line β_j^c remain the same.

Task 2

Assuming that our input data is mean-centered, enabling us to ignore the intercept term β_0. The posterior distribution is then described as follows:

$$Pr(\beta|y,X)=\frac{1}{K}PR(y|\beta,X)Pr(\beta)$$

where $K=K(y,X)=\int Pr(y|\beta,X)Pr(\beta)d\beta$. We can see that K does not depend on β. Under the specified assumptions, we can observe that

$$Pr(\beta|y,X)=\frac{1}{Z}\frac{1}{(2\pi)^{\frac{p}{2}}\sigma^2}exp\left(-\frac{(y-X\beta)^T(y-X\beta)}{2\sigma^2}\right)\frac{1}{(2\pi)^2\tau^p}exp\left(-\frac{\beta^T\beta}{2\tau^2}\right)$$

We can conclude that

$$log(Pr(\beta|y,X))=-W-\frac{(y-X\beta)^T(y-X\beta)}{2\sigma^2}-\frac{\beta^T\beta}{2\tau^2}$$

where D does not depend on β. To maximize this expression we can define $\hat{\beta}$ as

$$\hat{\beta} = \left(X^T X + \frac{\sigma^2}{\tau^2} I \right)^{-1} X^T y$$

When we set $\lambda = \frac{\sigma^2}{\tau^2}$, we can observe that the method mentioned above is essentially the same as ridge regression.

It is evident that the probability distribution $Pr(\beta|y, X)$ takes on a Gaussian form, with its mean and mode being in perfect alignment. We will proceed to demonstrate that the mean value is equivalent to $\hat{\beta}$. To do so, we should note that equation above for $Pr(\beta|y, X)$ implies that the covariance matrix Σ follows the relationship below

$$\Sigma^{-1} = \frac{1}{\sigma^2} \left(X^T X + \frac{\sigma^2}{\tau^2} I \right)$$

This equation leads to the conclusion that $\hat{\beta} = \frac{1}{\sigma^2} \Sigma X^T y$. When we equate the relevant elements in the equation above for $Pr(\beta|y, X)$, it becomes apparent that this expression serves as the mean value.

Task 3

Exercise 3: Show that in case the design matrix $\boldsymbol{X}$ of a linear regression model is orthonormal (i.e., $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$), that the estimators of β_j are given by the following equations with $\hat{\beta}_j$ denoting the ordinary least squares estimate:

(a)

1. Begin with the ordinary least squares (OLS) estimator for the entire vector of coefficients β when the design matrix $\boldsymbol{X}$ is orthonormal:

$$\hat{\beta} = \boldsymbol{X}^\top Y.$$

2. Now, consider the OLS estimator for a subset of size M:

$$\hat{\beta}_M = \boldsymbol{X}_M^\top Y,$$

where $\boldsymbol{X}_M$ is the submatrix of $\boldsymbol{X}$ containing the columns corresponding to the subset of size M.

3. Focus on the estimator for a single coefficient β_j. This can be expressed as:

$$\hat{\beta}_j = \boldsymbol{e}_j^\top \hat{\beta},$$

where $\boldsymbol{e}_j$ is the jth standard basis vector.

4. Using the expression for $\hat{\beta}$, rewrite the equation for $\hat{\beta}_j$:

$$\hat{\beta}_j = \boldsymbol{e}_j^\top \boldsymbol{X}^\top Y.$$

5. Since $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$, it follows that $\boldsymbol{X}^\top = \boldsymbol{X}^{-1}$.

6. With this understanding, simplify the equation for $\hat{\beta}_j$:

$$\hat{\beta}_j = \boldsymbol{e}_j^\top Y - \sum_{k \neq j} \boldsymbol{e}_j^\top \boldsymbol{X}_k^\top Y.$$

7. Due to the orthonormality of $\boldsymbol{X}$, it is known that $\boldsymbol{e}_j^\top \boldsymbol{X}_k = 0$ for $k \neq j$. As a result, the summation term becomes zero.

8. Therefore, it can be concluded that:

$$\hat{\beta}_j = \boldsymbol{e}_j^\top Y.$$

9. Finally, express the estimator for β_j in terms of the estimator for the best subset of size M:

$$\hat{\beta}_j = \hat{\beta}_j I\left(\left|\hat{\beta}_j\right| \geq \left|\hat{\beta}_M\right|\right),$$

where $I()$ denotes the indicator function, and $\hat{\beta}_j$ and $\hat{\beta}_M$ represent the OLS estimators for β_j and the best subset of size M .

Additionally, we are aware that choosing the M coefficients with the highest absolute values means we have the best subset in general. As a result, we may now employ the indicator function, which checks whether a coefficient's absolute value is $\geq$ equal to that of the H-th longest coefficient and then for the j -th coefficient we get the desired result.

(b)

Ridge with penalty λ :

$$\hat{\beta}_j/(1+\lambda).$$

When the design matrix $\boldsymbol{X}$ of a linear regression model is orthonormal, i.e., $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$, the estimators of β_j are given by the following equations with $\hat{\beta}_j$ denoting the ordinary least squares estimate:

b) Ridge with penalty λ :

$$\hat{\beta}_j/(1+\lambda).$$

To show this, we can start with the ordinary least squares estimate:

$$\hat{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

Since $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$, we have:

$$\hat{\beta} = \boldsymbol{X}^\top \boldsymbol{y}.$$

Now, for the ridge regression estimate, we add a penalty term to the least squares criterion:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda \|\beta\|^2 \right\}.$$

Using the orthonormality of $\boldsymbol{X}$, we can write:

$$\|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 = \|\boldsymbol{y}\|^2 - 2\boldsymbol{y}^\top \boldsymbol{X}\beta + \beta^\top \boldsymbol{X}^\top \boldsymbol{X}\beta = \|\boldsymbol{y}\|^2 - 2\boldsymbol{y}^\top \boldsymbol{X}\beta + \|\beta\|^2.$$

Substituting this into the ridge regression criterion, we get:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \|\boldsymbol{y}\|^2 - 2\boldsymbol{y}^\top \boldsymbol{X}\beta + \|\beta\|^2 + \lambda\|\beta\|^2 \right\}.$$

Taking the derivative with respect to β and setting it to zero, we get:

$$-2\boldsymbol{X}^\top \boldsymbol{y} + 2(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})\beta = 0.$$

Solving for β, we get:

$$\hat{\beta}_{ridge} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

Using the orthonormality of $\boldsymbol{X}$ again, we can simplify this to:

$$\hat{\beta}_{ridge} = \frac{\boldsymbol{X}^\top \boldsymbol{y}}{1 + \lambda}.$$

Therefore, the estimators of β_j for ridge regression with penalty λ are given by:

$$\hat{\beta}_{j,ridge} = \frac{\boldsymbol{x}_j^\top \boldsymbol{y}}{1 + \lambda},$$

where $\boldsymbol{x}_j$ is the jth column of $\boldsymbol{X}$.

(c)

1. Start with the Lasso optimization problem, which is a linear regression method using an L1-norm penalty term:
$$\min_{\boldsymbol{\beta}} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$
2. Recognize that when the design matrix $\boldsymbol{X}$ is orthonormal, the coefficients β_j can be estimated as follows:
$$\hat{\beta}_j = \boldsymbol{X}_j^\top \boldsymbol{y},$$
where $\boldsymbol{X}_j$ is the j-th column of the orthonormal matrix $\boldsymbol{X}$.
3. Simplify $\boldsymbol{X}_j^\top \boldsymbol{y}$ due to the orthonormality:
$$\boldsymbol{X}_j^\top \boldsymbol{y} = |\boldsymbol{X}_j^\top \boldsymbol{y}|.$$
4. Rewrite the Lasso optimization problem with this simplification and the explicit expression for $\hat{\beta}_j$:
$$\min_{\boldsymbol{\beta}} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\hat{\beta}_j|.$$
5. Continue the simplification by taking the absolute value out of the summation (since it's constant with respect to β_j) and replacing $|\boldsymbol{X}_j^\top \boldsymbol{y}|$ with $|\hat{\beta}_j|$:
$$\min_{\boldsymbol{\beta}} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\hat{\beta}_j|.$$
6. The problem is now in a form that is directly related to the Lasso equation with penalty λ. The goal is to minimize the same objective function as in the Lasso problem.

7. The solution to this optimization problem aligns with the Lasso equation with penalty λ:

$$\min_{\beta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda|\hat{\beta}_j| = \text{sign}(\hat{\beta}_j) \left(|\hat{\beta}_j| - \lambda \right)_+ .$$

By proving this equivalence, it is demonstrated that when the design matrix $\boldsymbol{X}$ of a linear regression model is orthonormal, the estimators of β_j are given by the Lasso equation with penalty λ:

$$\text{sign}(\hat{\beta}_j) \left(|\hat{\beta}_j| - \lambda \right)_+ .$$

Task 4

The pmf in the Poisson regression model is given by:

$$P(Y_i = y_i|\mu_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!},$$

where μ_i is the expected value of Y_i. Using the log-link function $g(\mu_i) = \log(\mu_i)$, we have:

$$\log(\mu_i) = \beta_0 + \beta_1 x_i$$

Therefore we get:

$$\mu_i = e^{\beta_0 + \beta_1 x_i}$$

So we get for the fitted means for group A ($x_i = 1$) and group B ($x_i = 0$):

$$\hat{\mu}_A = e^{\beta_0 + \beta_1}, \quad \hat{\mu}_B = e^{\beta_0}$$

Now lets look at the likelihood for the Poisson regression, which is given by:

$$L(\beta_0, \beta_1; \boldsymbol{y}, \boldsymbol{X}) = \prod_{i=1}^{n} \frac{e^{-e^{\beta_0 + \beta_1 x_i}} (e^{\beta_0 + \beta_1 x_i})^{y_i}}{y_i!}$$

Therefore the log-likelihood is:

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \left(y_i(\beta_0 + \beta_1 x_i) - e^{\beta_0 + \beta_1 x_i} - \log(y_i!) \right)$$

To find the maximum likelihood estimates, we need to solve the score equations:

$$\frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_0} = 0, \quad \frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_1} = 0.$$

Derivative with respect to β_0:

$$\frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_0} = \sum_{i=1}^{n} \left(y_i - e^{\beta_0 + \beta_1 x_i} \right) = 0$$

Solving this leads to:

$$\sum_{i=1}^{n_A} \left(y_i - e^{\beta_0 + \beta_1} \right) + \sum_{i=n_A+1}^{n_A+n_B} \left(y_i - e^{\beta_0} \right) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n_A} (y_i) + \sum_{i=n_A+1}^{n_A+n_B} (y_i) - n_A e^{\beta_0 + \beta_1} - (n_A + n_B - n_A)e^{\beta_0} = 0$$

$$\Leftrightarrow \sum_{i=1}^{n_A}(y_i) + \sum_{i=n_A+1}^{n_A+n_B}(y_i) - n_A e^{\beta_0+\beta_1} - n_B e^{\beta_0} = 0$$

The sample means of group A and group B are:

$$\bar{y}_A = \frac{1}{n_A}\sum_{i=1}^{n_A}(y_i), \quad \bar{y}_B = \frac{1}{n_B}\sum_{i=n_A+1}^{n_A+n_B}(y_i)$$

Therefore we get:

$$n_A\bar{y}_A + n_B\bar{y}_B - n_A e^{\beta_0+\beta_1} - n_B e^{\beta_0} = 0 \quad (1)$$

Next we look at the derivative with respect to β_1:

$$\frac{\partial \ell(\beta_0,\beta_1)}{\partial \beta_1} = \sum_{i=1}^{n}\left(y_i x_i - x_i e^{\beta_0+\beta_1 x_i}\right) = 0$$

Solving this we get:

$$\begin{aligned}
&\sum_{i=1}^{n_A}\left(y_i - e^{\beta_0+\beta_1}\right) = 0 \\
\Leftrightarrow \quad &\sum_{i=1}^{n_A}(y_i) - n_A e^{\beta_0+\beta_1} = 0 \\
\Leftrightarrow \quad &n_A\bar{y}_A - n_A e^{\beta_0+\beta_1} = 0 \\
\Leftrightarrow \quad &\bar{y}_A = e^{\beta_0+\beta_1}
\end{aligned}$$

Plugging that into (1) we get:

$$\begin{aligned}
&n_A\bar{y}_A + n_B\bar{y}_B - n_A e^{\beta_0+\beta_1} - n_B e^{\beta_0} = 0 \\
\Leftrightarrow \quad &n_A e^{\beta_0+\beta_1} + n_B\bar{y}_B - n_A e^{\beta_0+\beta_1} - n_B e^{\beta_0} = 0 \\
\Leftrightarrow \quad &\bar{y}_B = e^{\beta_0}
\end{aligned}$$

Thus we have shown that:

$$\hat{\mu}_A = e^{\beta_0+\beta_1} = \bar{y}_A, \quad \hat{\mu}_B = e^{\beta_0} = \bar{y}_B$$

Therefore, the fitted means $\hat{\mu}_A$ and $\hat{\mu}_B$ equal the sample means.

Task 5

a

Assume $Y \sim Bin(T,\pi)$. The pmf is given by:

$$P(X=y) = \binom{T}{y}\pi^y(1-\pi)^{T-y}$$

If we now take the logarithm and then the exponential again we get:

$$\begin{aligned}
&= \exp\left\{y\log(\pi) + (T-y)\log(1-\pi) + \log\binom{T}{y}\right\} \\
&= \exp\left\{y\log\left(\frac{\pi}{1-\pi}\right) + T\log(1-\pi) + \log\binom{T}{y}\right\}
\end{aligned}$$

Therefore we have that:

$$\theta = \log\left(\frac{\pi}{1-\pi}\right) \quad \Leftrightarrow \quad e^\theta = \frac{\pi}{1-\pi}$$
$$\Leftrightarrow \quad e^\theta(1-\pi) = \pi \quad \Leftrightarrow \quad e^\theta = \pi(1+e^\theta)$$
$$\Leftrightarrow \quad \pi = \frac{e^\theta}{1+e^\theta} \quad \Longrightarrow \quad 1-\pi = \frac{1+e^\theta-e^\theta}{1+e^\theta} = (1+e^\theta)^{-1}$$

Plugging this into the pmf we have:

$$\exp\left\{y\log\left(\frac{\pi}{1-\pi}\right) + T\log(1-\pi) + \log\binom{T}{y}\right\}$$
$$= \exp\left\{y\log\left(\frac{\pi}{1-\pi}\right) + T\log((1+e^\theta)^{-1}) + \log\binom{T}{y}\right\}$$
$$= \exp\left\{y\log\left(\frac{\pi}{1-\pi}\right) - T\log(1+e^\theta) + \log\binom{T}{y}\right\}$$

Therefore we get:

$$\theta = \log\left(\frac{\pi}{1-\pi}\right), \quad b(\theta) = T\log(1+e^\theta), \quad a(\phi) = 1, \quad \phi = 1, \quad c(y,\phi) = \log\binom{T}{y}$$

Therefore it can be written as a univariate exponential dispersion family.

b

We have that the logit link is $logit(p) = \log\frac{p}{1-p}$. With the logit-link we regress the logit of p on the explanatory variables. For example for a linear regression with one explanatory variable we have:

$$logit(p) = \log\frac{p}{1-p} = \beta_0 + \beta_1 x$$

If we want to calculate p from that, we get that

$$p = \frac{e^{logit(p)}}{1+e^{logit(p)}} = \frac{1}{1+e^{-logit(p)}}$$

The likelihood of of the binomial regression model with y_i successes out of T_i trials is the product of the individual probabilities:

$$L = \prod_{i=1}^{T}\binom{T_i}{y_i}p_i^{y_i}(1-p_i)^{T_i-y_i}$$

To get the log-likelihood we just take the logarithm:

$$\ell = \log(L) = \sum_{i=1}^{T}\left(\log\binom{T_i}{y_i} + y_i\log(p_i) + (T_i-y_i)\log(1-p_i)\right)$$

Now plugging in the exponent form for the p_i:

$$= \sum_{i=1}^{T}\left(\log\binom{T_i}{y_i} + y_i\log(\frac{1}{1+e^{-\beta^T x_i}}) + (T_i-y_i)\log(1-\frac{1}{1+e^{-\beta^T x_i}})\right)$$

$$= \sum_{i=1}^{T}\left(\log\binom{T_i}{y_i} + y_i\log(\frac{1}{1+e^{-\beta^T x_i}}) + (T_i-y_i)\log(\frac{e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}})\right)$$

$$= \sum_{i=1}^{T} \left(\log \binom{T_i}{y_i} + y_i \left[\log(\frac{1}{1+e^{-\beta^T x_i}}) - \log(\frac{e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}}) \right] + T_i \log(\frac{e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}}) \right)$$

$$= \sum_{i=1}^{T} \left(\log \binom{T_i}{y_i} + y_i \log(e^{\beta^T x_i}) + T_i \log(\frac{1}{1+e^{\beta^T x_i}}) \right)$$

$$= \sum_{i=1}^{T} \left(\log \binom{T_i}{y_i} + y_i \beta^T x_i - T_i \log(1+e^{\beta^T x_i}) \right)$$

c

For the score function we derive the log-likelihood function wrt. β:

$$\nabla_\beta \ell = \nabla_\beta \sum_{i=1}^{T} \left(\log \binom{T_i}{y_i} + y_i \beta^T x_i - T_i \log(1+e^{\beta^T x_i}) \right)$$

$$= \sum_{i=1}^{T} \nabla_\beta \left(\log \binom{T_i}{y_i} + y_i \beta^T x_i - T_i \log(1+e^{\beta^T x_i}) \right)$$

$$= \sum_{i=1}^{T} \nabla_\beta (y_i \beta^T x_i) - \nabla_\beta (T_i \log(1+e^{\beta^T x_i}))$$

$$= \sum_{i=1}^{T} y_i x_i - T_i \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}} x_i$$

$$= \sum_{i=1}^{T} y_i x_i - T_i \frac{1}{1+e^{-\beta^T x_i}} x_i$$

Plugging back in the probability:

$$= \sum_{i=1}^{T} y_i x_i - T_i p_i x_i$$

$$= \sum_{i=1}^{T} (y_i - T_i p_i) x_i$$

Therefore:

$$s(\beta) = \sum_{i=1}^{T} (y_i - T_i p_i) x_i$$

where $p_i = \frac{1}{1+e^{-\beta^T x_i}}$

Task 6

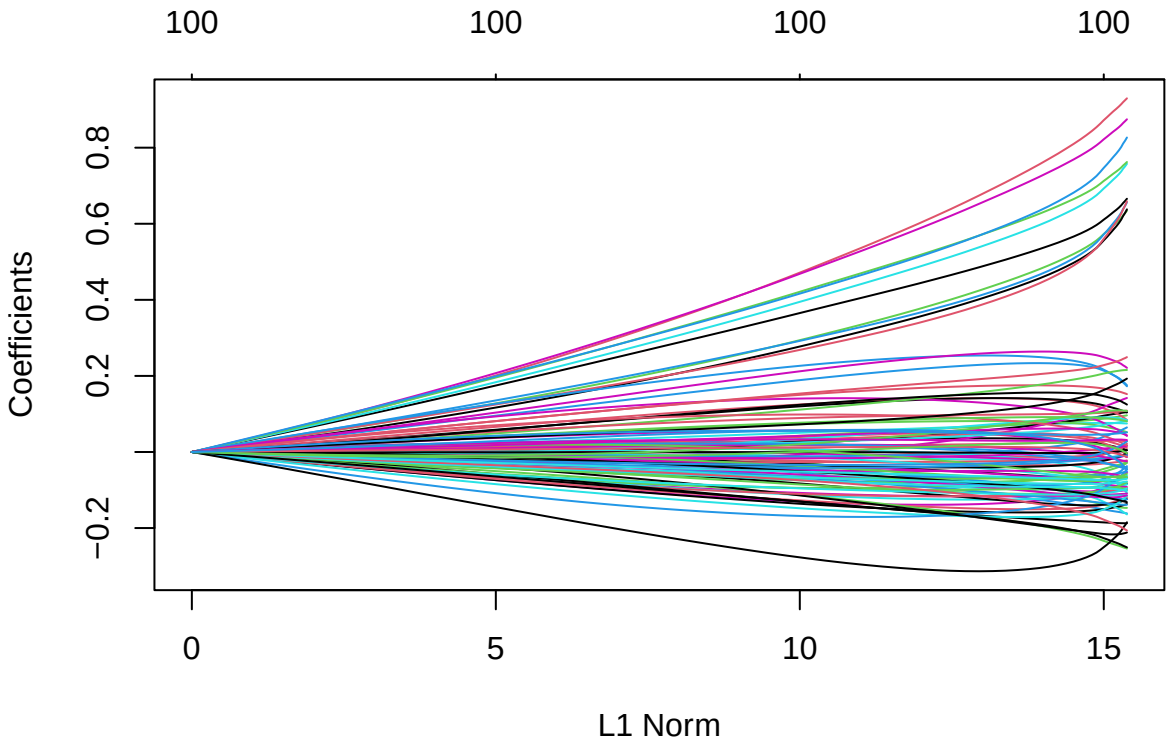
```
library(MASS)

set.seed(123)

# a) draw from standard multivariate normal
X <- mvrnorm(n = 100, mu = rep(0, 100), Sigma = diag(1, nrow = 100, ncol = 100))
names(X) <- paste0("X", 1:100)
```

```r
# b) use 10 first columns of X and simulated noise to get y
y <- apply(X[, 1:10], MARGIN = 1, FUN = sum) + rnorm(100, mean = 0, sd = sqrt(0.01))

# c) fit LASSO and ridge models with different values of lambda

library(glmnet)
```

```
## Warning: Paket 'glmnet' wurde unter R Version 4.2.3 erstellt

## Lade nötiges Paket: Matrix

## Warning: Paket 'Matrix' wurde unter R Version 4.2.3 erstellt

## Loaded glmnet 4.1-8
```

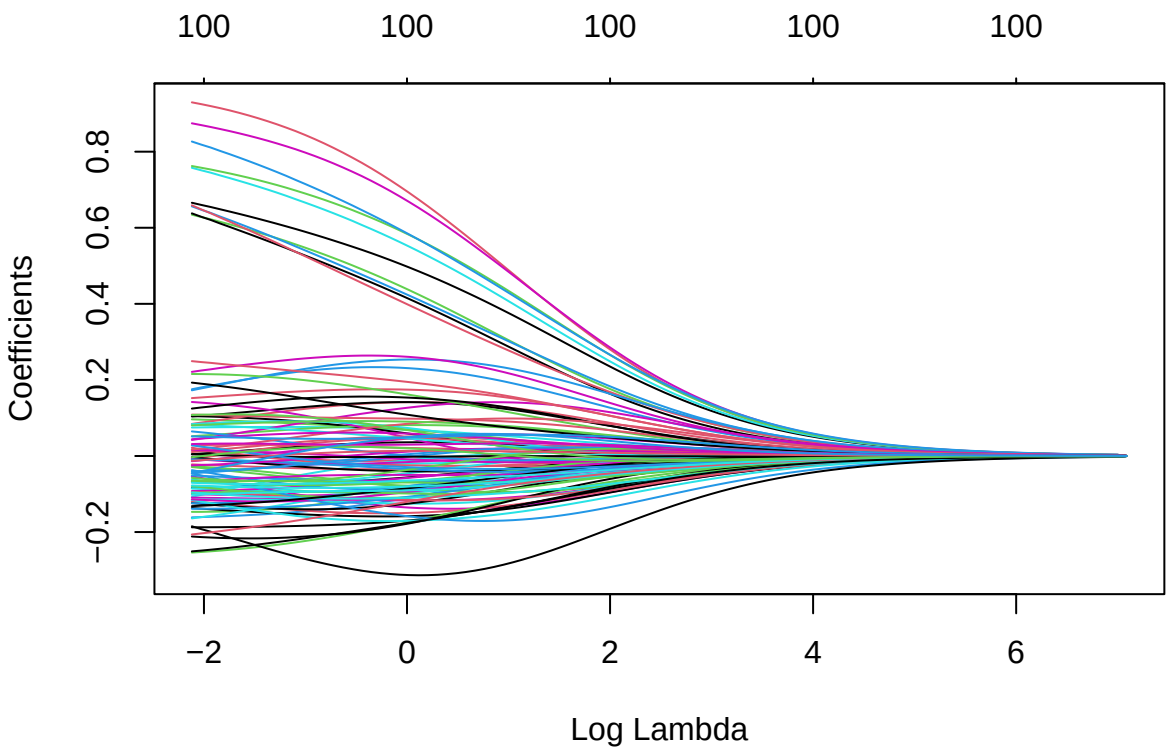
```r
# alpha = 0 is RIDGE, alpha = 1 is LASSO according to elastic net mixing

# glmnet chooses suitable lambda values itself but one could also specify the
# lambda sequence manually (argument: lambda = sequence). We go with the implemented selection of lambd
# we expect sensible behavior.

RIDGE <- glmnet(x = X, y = y, alpha = 0)

LASSO <- glmnet(x = X, y = y, alpha = 1)
```

```r
# d) plot for RIDGE
plot(RIDGE)
```



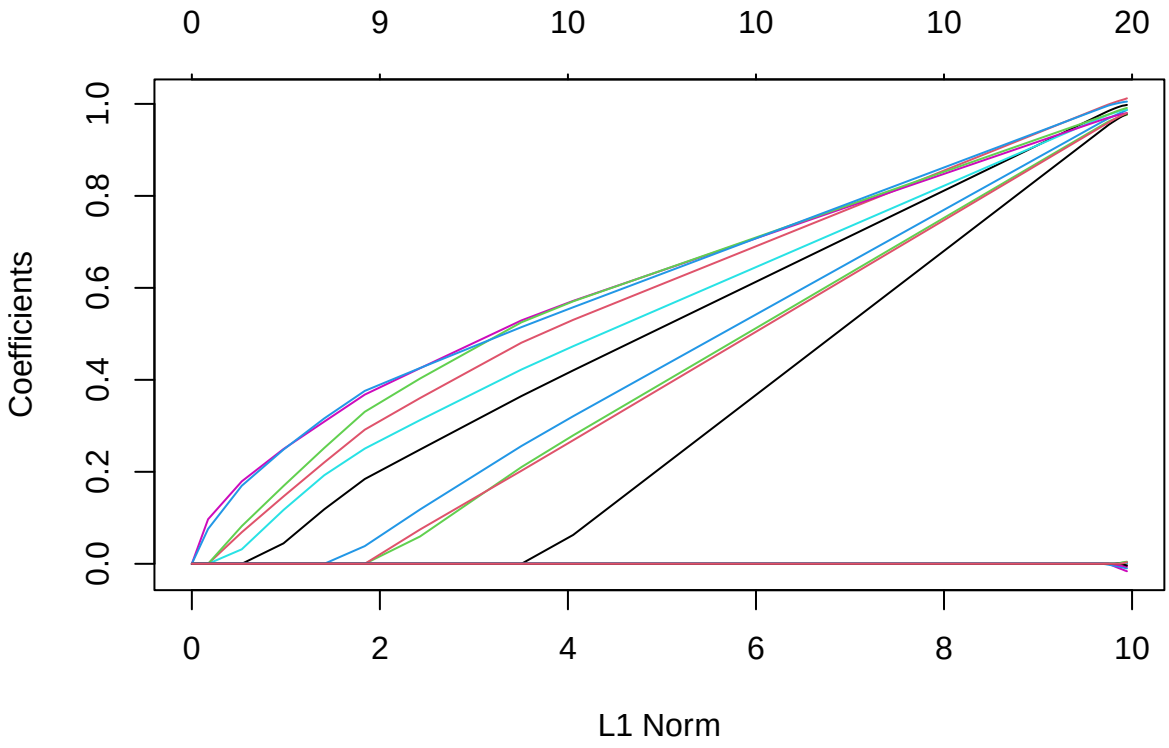
The default plot for Ridge-Regression shows on the lower x-axis the L^1-norm of the whole coefficient vector and on the upper x-axis the number of non-zero coefficients dependent on the current value of λ. The y-axis represents the values of the coefficients and each line corresponds to a variable. Thus the curves show the values of the coefficients against the L^1-norm at specific λ. For Ridge-Regression we have that the number of non-zero coefficients does not change, i.e. it stays at $n = 100$ as $X^{n\times n}$. As for $\lambda \to \infty$ all coefficients shrink to 0 and we end up with the null model, thus there the L^1 norm of the coefficient vector will be 0 too. Hence the interpretation of the plot is as follows: with $\lambda \to 0$ we observe larger coefficients in magnitude and thus also larger L^1 norm of the coefficient vector. We also observe that with low λ a small subset of coefficients is much larger in magnitude than the majority of coefficients which is owed to the fact that we used only the first ten variables in $X^{n\times n}$ to construct y. With increasing λ all coefficients shrink to 0 but the ones asssociated with the "more important" variables (by magnitude) shrink slower as they vary more with y than the other variables.

```
# d) plot with xvar = lambda
plot(RIDGE, xvar = "lambda")
```



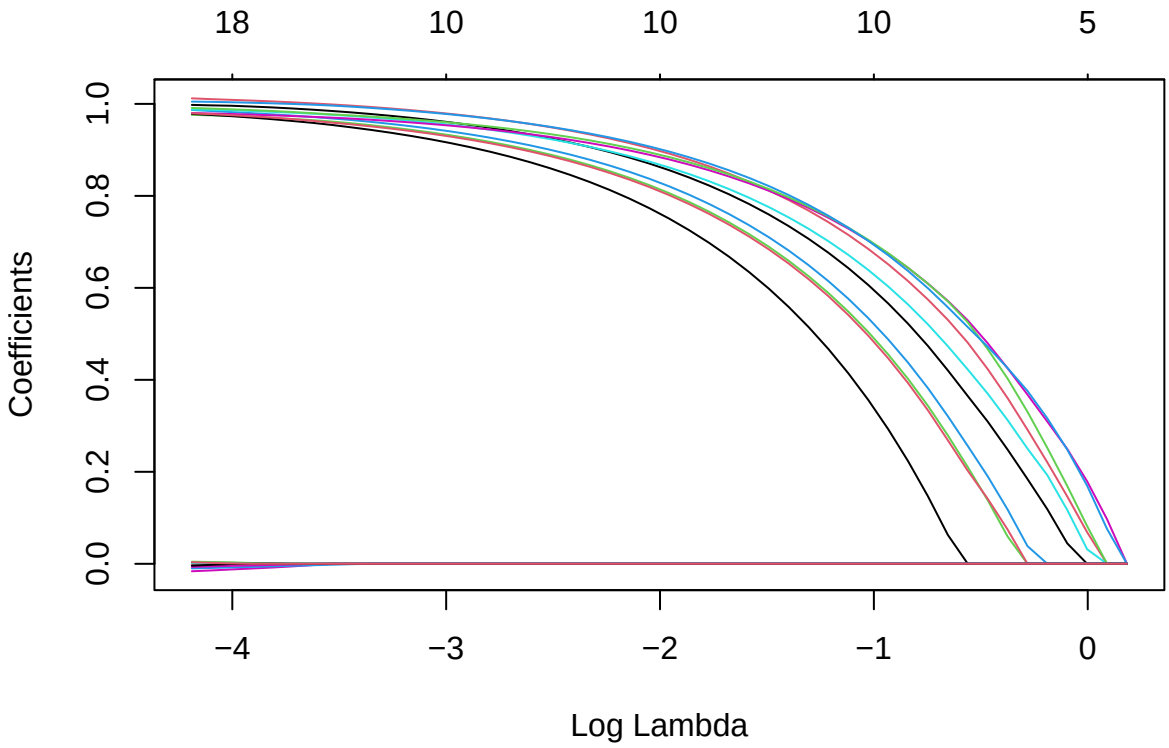
In comparison we see in the plot with $log(\lambda)$ on the x-Axis how coefficients shrink with Lambda. Again we see that in the beginning, where we are closer to the OLS solution, a small subset of variables has higher coefficients, while all coefficients shrink to 0 as $\lambda \to \infty$.

```
# d) plot for LASSO
plot(LASSO)
```



For LASSO, the default plot gives us the same quantities on the axes. However, we immediately see, that the number of non-zero coefficients is not constant as in RIDGE regression but shrinking as λ increases (and thus the L^1 norm decreases). Already at the highest value for the L^1 norm LASSO set already 80/100 coefficients to 0. With $\lambda \to \infty \Rightarrow L^1$-norm $\to 0$ more and more coefficients are shrunk to 0.

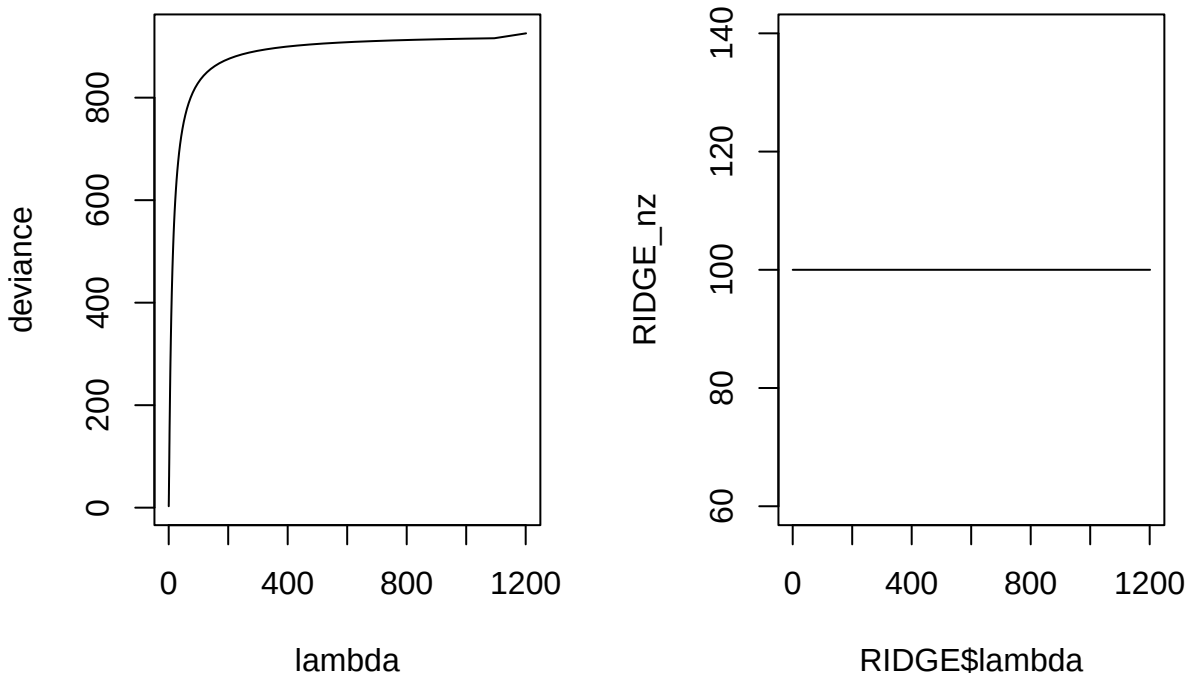
```
# d) plot with xvar = lambda
plot(LASSO, xvar = "lambda")
```



If we put $log(\lambda)$ on the x-axis we observe in essence the same as in the default plot, also in comparison to RIDGE. For low λ we see a subset of all variables having non-zero coefficients. With λ increasing we observe more and more coefficients becoming 0 until we end up with the null-model again.

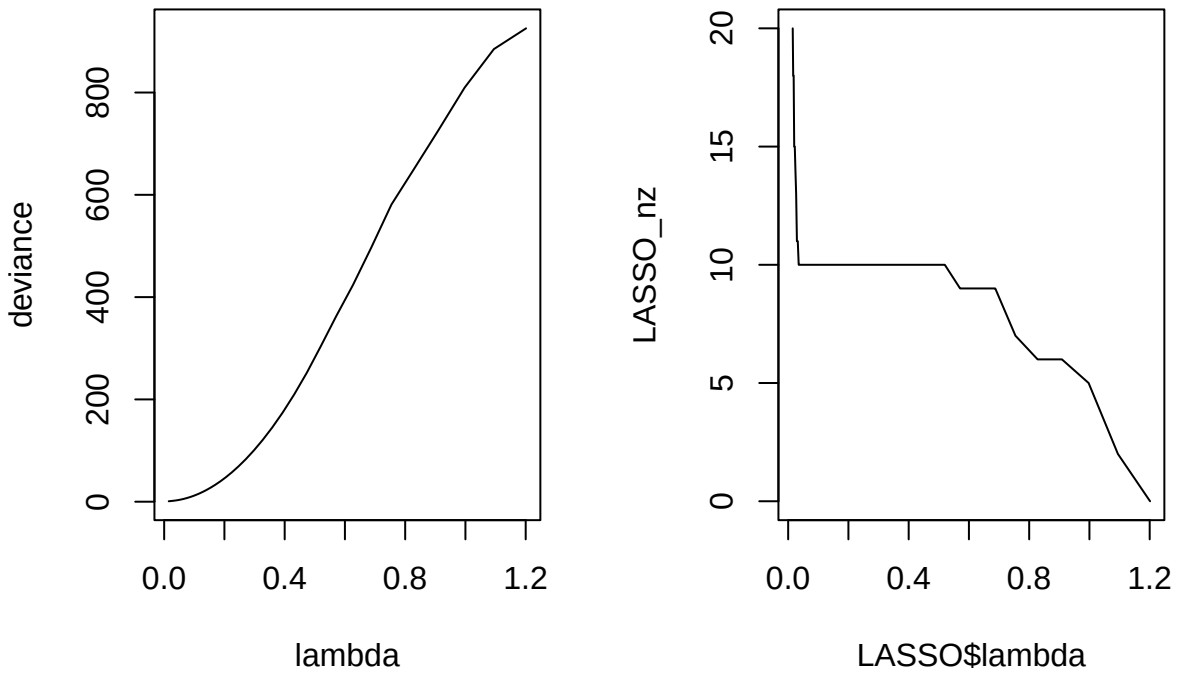
```
# get deviance and lambda
RIDGE_dev_l <- data.frame(deviance = deviance(RIDGE), lambda = RIDGE$lambda)
LASSO_dev_l <- data.frame(deviance = deviance(LASSO), lambda = LASSO$lambda)

# get number of non-zero coefficients and lambda
RIDGE_nz <- unlist(lapply(predict(RIDGE, type = "nonzero"), length))
LASSO_nz <- unlist(lapply(predict(LASSO, type = "nonzero"), length))
```

```
{
  par(mfrow = c(1, 2))
# plot deviance in dependence of lambda for RIDGE
with(RIDGE_dev_l, plot(x = lambda, y = deviance, type = "l"))
# plot number of non-zeros in dependence of lambda for RIDGE
plot(x = RIDGE$lambda, y = RIDGE_nz, type = "l")
}
```



We can see that for RIDGE regression the deviance is sharply increasing with λ until it reaches a point where increases become smaller and smaller. As already pointed out the number of non-zero coefficient always stays at 100 for RIDGE. The losses in accuracy of fitted values is simply owed to the fact that all coefficients shrink towards 0 letting the fitted values become more and more similar to the intercept which is not subject to shrinkage. This consequently lets the sum of squared residuals increase.

```
{
  par(mfrow = c(1, 2))
# plot deviance in dependence of lambda for RIDGE
with(LASSO_dev_l, plot(x = lambda, y = deviance, type = "l"))
# plot number of non-zeros in dependence of lambda for RIDGE
plot(x = LASSO$lambda, y = LASSO_nz, type = "l")
}
```



For LASSO we see that deviance is also increasing in λ but the increase is slower than for RIDGE. Looking at the number of non-zero coefficients in dependence of λ we can observe that more and more coefficients shrink to 0 as λ increases. However it is expected that the coefficients of the variables with explanatory power shrink slower than those of variables that do not vary with y so much. As thus variables vanish from the model that are less important, the losses in terms of RSS are not so severe as for RIDGE regression.

Task 7

We have that $G \sim Multinomial(\boldsymbol{\pi})$ with $\boldsymbol{\pi}$ being the vector of probabilities for class membership. We further know that x given G is multivariate normally distributed with μ_k, Σ_k given $G = k$.

Log-odds are given by

$$log\frac{P(G=k|x)}{P(G=l|x)}$$

and we find $P(G=k|x) = \frac{P(x|G=k)P(G=k)}{P(x)}$ by Bayes theorem. As $P(X)$ drops, log odds reduce to

$$log\frac{P(G=k|x)}{P(G=l|x)} = log\frac{P(x|G=k)\pi_k}{P(x|G=l)\pi_l} = log(\pi_k) - log(\pi_l) + log(P(x|G=k)) - log(P(x|G=l))$$

we now just have to show that this is a quadratic function in x.

We know that $P(x|G=k)$ and $P(x|G=l)$ are normal densities and that only the associated terms are relevant hence we focus on them.

Taking the difference of the logs of the densities into account we get

$$\frac{1}{2}\left[(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)-(x-\mu_l)'\Sigma_k^{-1}(x-\mu_l)\right]$$

We can further ignore the common factor 1/2 and expand to

$$x'\Sigma_k^{-1}x+\mu_k'\Sigma_k^{-1}\mu_k-2\mu_k'\Sigma_k^{-1}x-x'\Sigma_l^{-1}x-\mu_l'\Sigma_l^{-1}\mu_l+2\mu_l'\Sigma_l^{-1}x.$$

Consolidating terms yields

$$x'\left(\Sigma_k^{-1}-\Sigma_l^{-1}\right)x-2\left(\mu_k'\Sigma_k^{-1}-\mu_l'\Sigma_l^{-1}\right)x+\left(\mu_k'\Sigma_k^{-1}\mu_k-\mu_l'\Sigma_l^{-1}\mu_l\right)$$

Let now $A=\Sigma_k^{-1}-\Sigma_l^{-1}$, $B=\mu_k'\Sigma_k^{-1}-\mu_l'\Sigma_l^{-1}$ and $C=\mu_k'\Sigma_k^{-1}\mu_k-\mu_l'\Sigma_l^{-1}\mu_l$. Then the above is of the form $x'Ax-2Bx+C$ which is a quadratic equation. As $log(\pi_k)$ and $log(\pi_l)$ would only add to C, we established that log odds are indeed a quadratic function of x.

It suffices to look at

$$x'\left(\Sigma^{-1}-\Sigma^{-1}\right)x-2\left(\mu_k'\Sigma^{-1}-\mu_l'\Sigma^{-1}\right)x+\left(\mu_k'\Sigma^{-1}\mu_k-\mu_l'\Sigma^{-1}\mu_l\right)$$

to observe that in case of equal variance-covariance matrices $A=0$ and hence we have $Bx+C$ which is linear in x.

For the univariate case, log odds LO are given by

$$\begin{aligned}LO&=\log(\pi_k)-\log(\pi_l)-\frac{1}{2\sigma^2}(x-\mu_k)^2+\frac{1}{2\sigma^2}(x-\mu_l)^2\\&=\log(\pi_k)-\log(\pi_l)-\frac{1}{2\sigma^2}\left(x^2-2\mu_{kx}+\mu_k^2-x^2+2\mu_l x-\mu_l^2\right)\\&=\log(\pi_k)-\log(\pi_l)-\frac{1}{2\sigma^2}\left(2\mu_l x-2\mu_k x+\mu_k^2-\mu_l^2\right)\end{aligned}$$

Note that the scaling factor $\frac{1}{\sigma\sqrt{2\pi}}$ cancels as already in the multivariate case when the variance-covariance matrix was equal. To see how LO changes when σ^2,μ_k,π_k change we have to look at first derivatives. For π_k we find

$$LO_{\pi_k}=\frac{1}{\pi_k}>0 \text{ as } \pi_k\geq 0,$$

for increases in σ^2 we get

$$LO_{\sigma^2}=-\frac{1}{\sigma^3}\left(2\mu_l x-2\mu_k x+\mu_k^2-\mu_l^2\right)$$

where the sign depends on the difference between the means and x and for an increase in μ_k

$$LO_{\mu_k}=-\frac{1}{\sigma^2}(\mu_k-x)$$

where again the sign is not evident as it depends on the difference between the mean and the random variable x.

Task 8

a)

We have to calculate x such that $P(G = 1|x) = P(G = 2|x)$. By Bayes' theorem we get for $G = i$, $i \in \{1, 2\}$

$$P(G = i|x) = \frac{P(x|G = i)P(G = 1)}{P(X)}$$

As $P(G = 1) = P(G = 2)$ we are left with the condition $P(x|G = 1) = P(x|G = 2)$. But we know that these are just the conditional distributions of x given in the task. Hence we can now equate the log densities of the standard normal and the normal distribution to obtain an expression for the value(s) of x fulfilling the initial equation. By equating densities we are left with

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \Leftrightarrow e^{-\frac{1}{2}x^2} = \sigma^{-1} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \Leftrightarrow -\frac{1}{2}x^2 = -log(\sigma) - \frac{1}{2\sigma^2}(x-\mu)^2$$

Solving for x yields then the optimal decision boundaries. Note that these are just the points where the density functions intersect, hence it is possible to have more than one boundary as we will see in b).

b)

Solving for x by plugging in μ, σ we get for a)

$$-\frac{1}{2}x^2 = -log(\sqrt{2}) - \frac{1}{4}x^2 \Rightarrow x = \pm 2\sqrt{log(\sqrt{2})}$$

and for b)

$$-\frac{1}{2}x^2 = \frac{1}{2}(x-1)^2 \Rightarrow x = 0.5$$

We can then calculate the error rate for a) by

$$\text{Bayes Rate} = \frac{1}{2}\left[\int_{-\infty}^{-2\sqrt{log(\sqrt{2})}} \phi_{0,1}(x)dx + \int_{-2\sqrt{log(\sqrt{2})}}^{2\sqrt{log(\sqrt{2})}} \phi_{0,2}(x)dx + \int_{2\sqrt{log(\sqrt{2})}}^{\infty} \phi_{0,1}(x)dx\right]$$

and for b) by

$$\text{Bayes Rate} = \frac{1}{2}\left[\int_{-\infty}^{0.5} \phi_{1,1}(x)dx + \int_{0.5}^{\infty} \phi_{0,1}(x)dx\right]$$

where we abuse the ϕ as the usual symbol for the standard normal density to express the normal density with corresponding μ, σ^2 and write ϕ_{μ,σ^2}. Further, $P(G = 1) = P(G = 2) = \frac{1}{2}$ can be pulled out of the integral in all of the cases. The respective error rates can then be found as below:

```
b_02 <- 2 * sqrt(log(sqrt(2)))

b_11 <- 0.5

# a) the bayes rate for N(0, 2) is
0.5 * pnorm(-b_02, 0, 1) + 0.5* integrate(dnorm, lower = -b_02, upper = b_02, mean = 0, sd = sqrt(2))$va
```

```
## [1] 0.416968
```

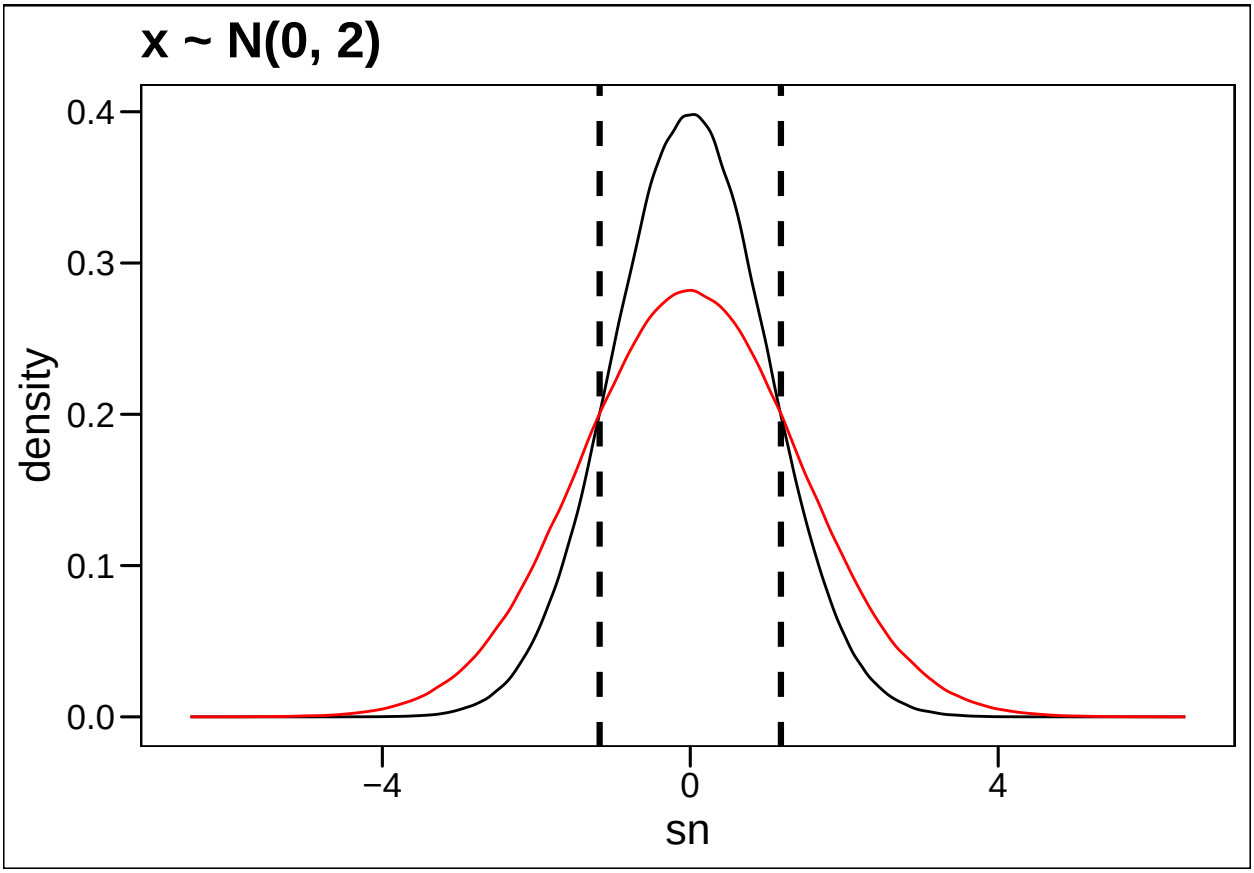
```
# b) the bayes rate for N(1, 1) is
0.5 * pnorm(b_11, 1, 1) + 0.5 * pnorm(b_11, 0, 1, lower.tail = F)
```

```
## [1] 0.3085375
```

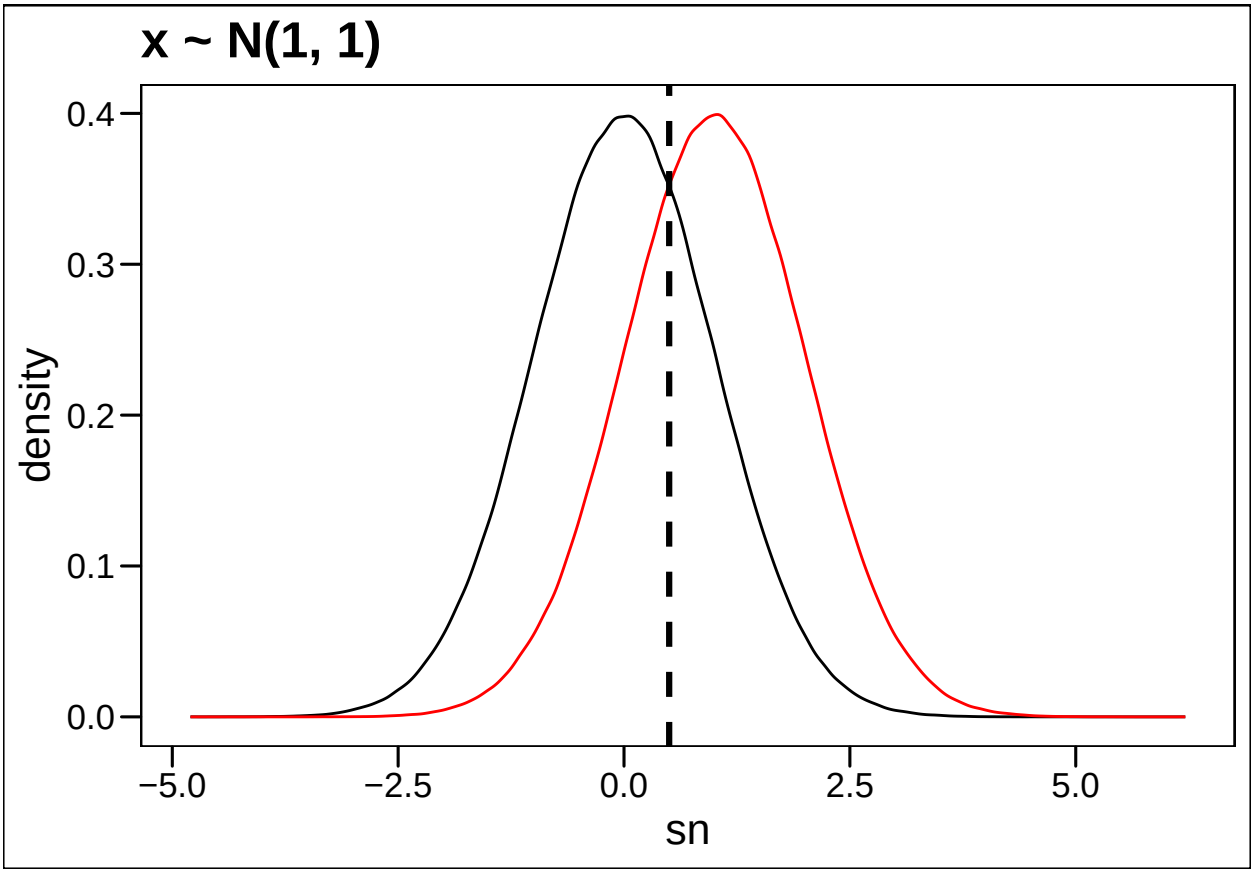
c)

If we visualize the densities of the distributions in part b) we can see that the error rate relates to the intersection of the areas under curves, as there it is uncertain to which class the observation with particular value x belongs.

```
library(tidyverse)
library(ggthemes)

# draw from standard normal
n <- 1000000
sn <- rnorm(n)

# draw from mu = 0, sigma2 = 2
n02 <- rnorm(n, mean = 0, sd = sqrt(2))

# draw from mu = 1, sigma2 = 1
n11 <- rnorm(n, mean = 1, sd = 1)

df <- data.frame(sn = sn, n02 = n02, n11 = n11) %>%
  pivot_longer(cols = c(n02, n11))

# plot the densities (boundaries have to be inserted but not so easy with one
# ggplot, maybe make two. Then it is just adding to vertical lines.)
ggplot() +
  geom_density(aes(x = sn)) +
  geom_density(aes(x = n02), color = "red") +
  geom_vline(xintercept = b_02, linetype = "dashed", size = 1.1) +
  geom_vline(xintercept = -b_02, linetype = "dashed", size = 1.1) +
  labs(title = "x ~ N(0, 2)") +
  theme_base()
```




```
ggplot() +
  geom_density(aes(x = sn)) +
  geom_density(aes(x = n11), color = "red") +
  geom_vline(xintercept = b_11, linetype = "dashed", size = 1.1) +
  labs(title = "x ~ N(1, 1)") +
  theme_base()
```



Task 9

Exercise 9

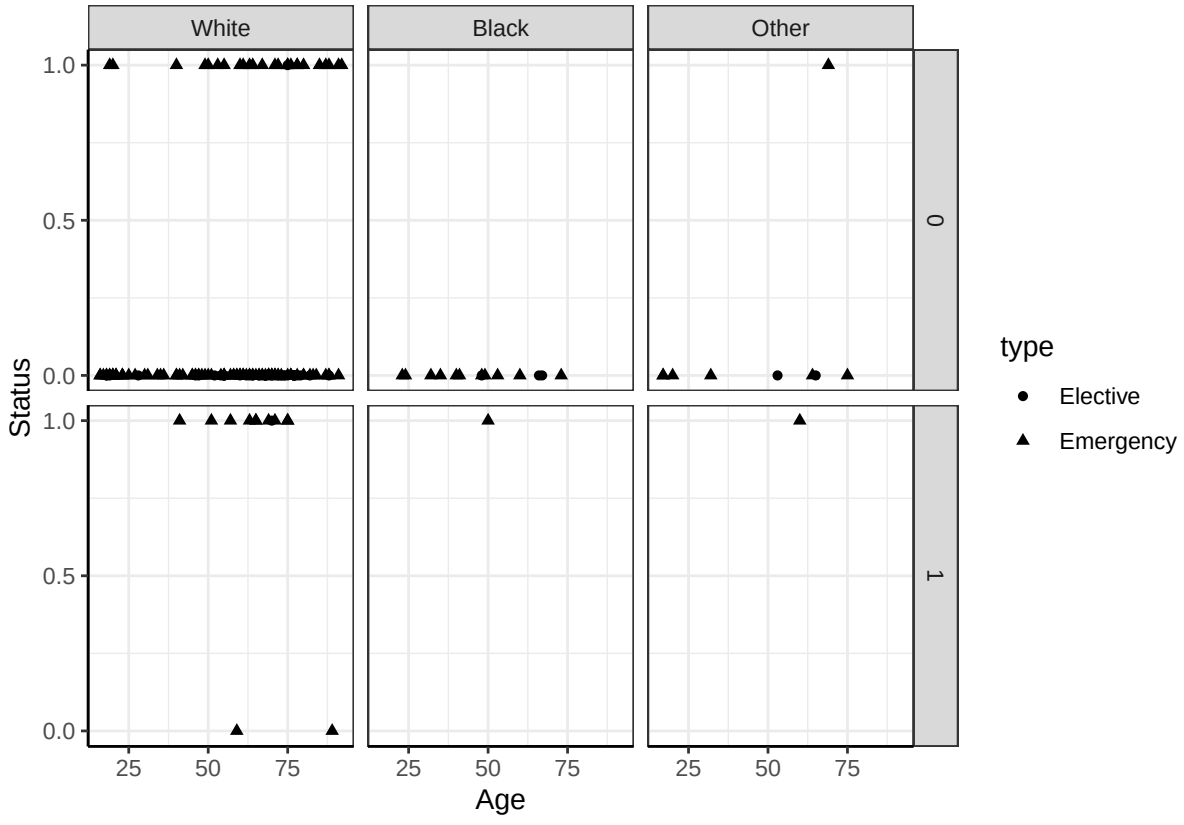
In this exercise, we delve deeper into common problems we might encounter when fitting a logistic regression model for the prediction of categorical data. The problem mentioned here consists of Complete or Quasi-Complete Separation, where the issue is not really connected to the model correctness or specification itself, but it is mainly due to "thin" data. This can be seen, e.g., in connection with really skewed distribution in the response variable. In our case, we observe that, fortunately, only in 20% of the reported cases the patient dies, for a sample size of 200. This is of course more relevant when also some categorical variables levels are not as frequent.

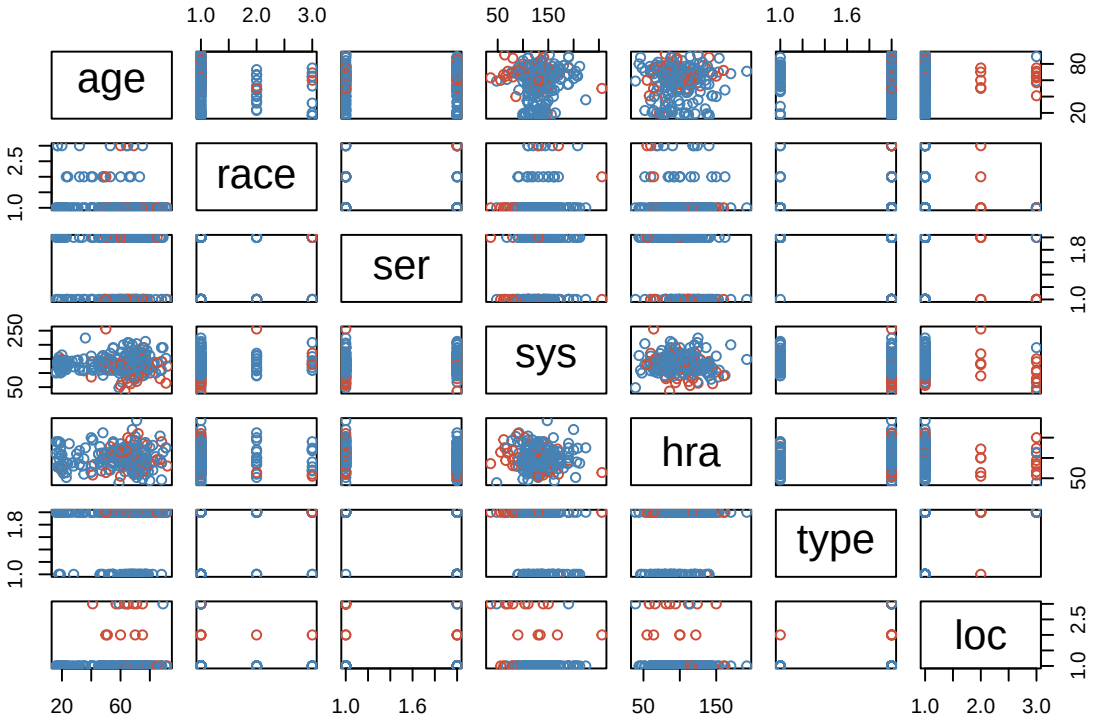
Point A

```
######################################################################
## A -  Fit a logistic regression model with all regressors
######################################################################
## A.1 Short exploratory Data Analysis

# Dataframe
df <- icu

# Variables classes
```

```r
#unlist(lapply(df, class))

# NAs
#unlist(lapply(df, function(x) sum(is.na(x))))

# Change variables of interest into numeric, binarize variables
df <- within(df,
             {
               # Response variable
               sta <- ifelse(sta == "Died", 1, 0)
               loc_bin <- ifelse(loc == "Nothing", 0, 1)
               race_bin <- ifelse(race == "White", 1, 0)
             })

## A.2 Model Fitting and summary. Note locComa and Stupor are both negative,
## but might have same pred power.


m1 <- glm(sta ~., family = binomial(link = "logit"), data = df[, 2:(length(df)-2)])
coef <- m1$coefficients
```

After running the logistic regression on all the variables we encounter the warning about fitted probabilities being numerically 0 or 1, which is a first sign of **separation**. To graphically illustrate the issue, we can look into the data with a grid plot, taking into account the loc, age and race variables. It is clear that issues of quasi-complete separation occur here, or complete separation if we had taken into account interaction terms. In the graph, the 0 and 1 binary classification reports the binarized loc variable, where 1 stands for the patient having Stupor or Coma reported.





In the summary we that the coefficients for locStupor, Black race, Emergency and other regressors are huge, indicating the maximum likelihood estimates do not exist. We can also notice the standard errors of these coefficients to be particularly big in magnitude.

```
##
## Call:
## glm(formula = sta ~ ., family = binomial(link = "logit"), data = df[,
##     2:(length(df) - 2)])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.50525  -0.53717  -0.17867  -0.00019   3.01708
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.548e+00  2.271e+00  -2.444  0.01454 *
## age            5.645e-02  1.848e-02   3.055  0.00225 **
## genderFemale  -7.215e-01  5.460e-01  -1.321  0.18639
## raceBlack     -1.617e+01  1.314e+03  -0.012  0.99018
## raceOther      5.829e-01  1.313e+00   0.444  0.65696
## serSurgical   -6.739e-01  6.289e-01  -1.071  0.28398
## canYes         3.483e+00  1.121e+00   3.106  0.00189 **
## crnYes         1.191e-01  8.449e-01   0.141  0.88786
## infYes        -1.081e-01  5.557e-01  -0.195  0.84573
## cprYes         1.032e+00  9.901e-01   1.043  0.29714
## sys           -2.084e-02  9.443e-03  -2.207  0.02732 *
## hra           -2.915e-03  1.032e-02  -0.282  0.77761
```

```
## preYes         1.279e+00  7.022e-01   1.822  0.06842 .
## typeEmergency  3.748e+00  1.342e+00   2.792  0.00523 **
## fraYes         1.649e+00  1.093e+00   1.509  0.13139
## po2<= 60      -6.765e-01  9.402e-01  -0.720  0.47179
## ph< 7.25       1.771e+00  1.212e+00   1.461  0.14410
## pco> 45       -2.084e+00  1.165e+00  -1.789  0.07361 .
## bic< 18       -2.623e-01  8.967e-01  -0.293  0.76985
## cre> 2.0       1.004e-01  1.131e+00   0.089  0.92925
## locStupor      3.771e+01  2.487e+03   0.015  0.98790
## locComa        3.458e+00  1.342e+00   2.578  0.00994 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 200.16  on 199  degrees of freedom
## Residual deviance: 112.17  on 178  degrees of freedom
## AIC: 156.17
##
## Number of Fisher Scoring iterations: 17
```

```
##     locStupor     raceBlack   (Intercept) typeEmergency        canYes
##  37.705271929  16.174546113   5.548262182   3.747879245   3.482602382
##       locComa       pco> 45      ph< 7.25        fraYes        preYes
##   3.458374688   2.083580267   1.770978482   1.649453062   1.279496861
##        cprYes  genderFemale      po2<= 60   serSurgical     raceOther
##   1.032231804   0.721456637   0.676510682   0.673862547   0.582918678
##       bic< 18        crnYes        infYes      cre> 2.0           age
##   0.262346172   0.119137902   0.108119860   0.100402678   0.056452457
##           sys           hra
##   0.020839745   0.002915162
```

Point B

So, as we observed in the previous point, we can absolutely make a case for a quasi-complete separation problem. Therefore, we start by pooling stratified variable levels into fewer ones.

```
##     gender1     gender2       race1       race2       race3        ser1
##      "Male"    "Female"     "White"     "Black"     "Other"   "Medical"
##        ser2        can1        can2        crn1        crn2        inf1
##  "Surgical"        "No"       "Yes"        "No"       "Yes"        "No"
##        inf2        cpr1        cpr2        pre1        pre2       type1
##       "Yes"        "No"       "Yes"        "No"       "Yes"  "Elective"
##       type2        fra1        fra2        po21        po22         ph1
## "Emergency"        "No"       "Yes"      "> 60"     "<= 60"   ">= 7.25"
##         ph2        pco1        pco2        bic1        bic2        cre1
##    "< 7.25"     "<= 45"      "> 45"     ">= 18"      "< 18"    "<= 2.0"
##        cre2        loc1        loc2        loc3
##     "> 2.0"   "Nothing"    "Stupor"      "Coma"
```

From the listing above we see that only two factors, namely **race** and **loc** have more than 2 levels, which were already binarized in the beginning. By refitting the full model, we see that pooling the levels together improves the estimation of the coefficients. At this point, one could turn some of the categorical variables

into continuous variables (1), or select a subset of the independent variables (2). The second option will introduce bias ,of course, but this seems to be the only available solution given the fact that we do not have the "original" measures for the syntetic categorical variables. We also note that with interaction terms, this problem would be even more accentuated, and we would not be able to determine whether such interactions would be relevant given the complete separation problem.

```
##
## Call:
## glm(formula = sta ~ ., family = binomial(link = "logit"), data = df2[,
##     2:length(df2)])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.80396  -0.56064  -0.20440  -0.08635   2.97729
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.086979   2.260515  -2.693  0.00709 **
## age            0.056393   0.018624   3.028  0.00246 **
## genderFemale  -0.639725   0.531393  -1.204  0.22864
## serSurgical   -0.673522   0.601902  -1.119  0.26315
## canYes         3.107051   1.045846   2.971  0.00297 **
## crnYes        -0.035708   0.801647  -0.045  0.96447
## infYes        -0.204933   0.553191  -0.370  0.71104
## cprYes         1.053483   1.006614   1.047  0.29530
## sys           -0.015472   0.008497  -1.821  0.06864 .
## hra           -0.002769   0.009607  -0.288  0.77317
## preYes         1.131942   0.671450   1.686  0.09183 .
## typeEmergency  3.079583   1.081584   2.847  0.00441 **
## fraYes         1.411402   1.029705   1.371  0.17047
## po2<= 60       0.073822   0.857044   0.086  0.93136
## ph< 7.25       2.354078   1.208804   1.947  0.05148 .
## pco> 45       -3.018442   1.253448  -2.408  0.01604 *
## bic< 18       -0.709284   0.909777  -0.780  0.43561
## cre> 2.0       0.295143   1.116925   0.264  0.79159
## race_bin       0.565729   0.926828   0.610  0.54160
## loc_bin        5.232292   1.226303   4.267 1.98e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 200.16  on 199  degrees of freedom
## Residual deviance: 120.78  on 180  degrees of freedom
## AIC: 160.78
##
## Number of Fisher Scoring iterations: 6
```

Point D

Now we want to improve the base model and the separation problem by selecting a subset of regressors using stepwise procedures based on AIC and BIC. We fit backward stepwise model selection.

The first based on AIC gives out:

$$sta \sim age + can + sys + type + ph + pco + loc_bin$$
$$AIC = 144.4$$

For the backward stepwise model selection absed on BIC, we have:

$$sta \sim age + can + type + loc_bin$$
$$AIC = 165.63$$

```
s1_AIC<- step(m2, direction = "backward", k = 2)
n <- dim(df2)[1]
s1_BIC <- step(m2, direction = "backward", k = log(n))
```

Point E

Now that we have obtained our models, we want to compare the full model with the stepwise selected models by benchmarking their log-likelihoods ant the in-sample misclassification.

```
boundary <- function(prob_vec, actual_vec){
  boundary_vec <- seq(0.01, 1, 0.01)

  mis_tmp <- 1
  for(i in boundary_vec){

    pred_tmp <- ifelse(prob_vec > i, 1, 0)
    mis_tmp <- ifelse(mean(pred_tmp != actual_vec) < mis_tmp, mean(pred_tmp != actual_vec), mis_tmp)

  }


}
pred_AIC <- ifelse(s1_AIC$fitted.values > 0.5, 1, 0)
pred_BIC <- ifelse(s1_BIC$fitted.values > 0.5, 1, 0)
pred_full <- ifelse(m2$fitted.values > 0.5, 1, 0)

## Misclassification errors
mis_AIC <- mean(pred_AIC != df2$sta)
mis_BIC <- mean(pred_BIC != df2$sta)
mis_FULL <- mean(pred_full != df2$sta)
results_mis <- c(mis_AIC, mis_BIC, mis_FULL)

## Loglikelihoods
lik_AIC <- logLik(s1_AIC)
lik_BIC <- logLik(s1_BIC)
lik_FULL <- logLik(m2)
results_lik <- c(lik_AIC, lik_BIC, lik_FULL)

## Matrix
vec <- c(results_mis, results_lik)
result_mat <- matrix(vec, ncol = 3, byrow = T)
rownames(result_mat) <- c("Misclassification", "Log-Lik")
```

```
colnames(result_mat) <- c("AIC", "BIC", "FULL")
result_mat
```

```
##                       AIC       BIC      FULL
## Misclassification   0.13   0.14000   0.11000
## Log-Lik           -64.22 -69.56732 -60.38917
```

From these results, we can see that the full model performs better in both metrics. However we need to keep in mind that this is only true in-sample, and a thorough analysis should be carried out also out-of-sample in such cases. This is also expected, since by eliminating covariates, we increase bias to reduce possible variance out-of-sample.

Task 10

```
suppressMessages(library("ISLR2"))
suppressMessages(library("ggplot2"))
suppressMessages(library("gridExtra"))
suppressMessages(library("dplyr"))
suppressMessages(library("magrittr"))

set.seed(7362)

data("Default", package="ISLR2")
df=Default

#First, we need to convert the 'default' and 'student' variables into the
#binary format. "Yes"=1, "No"=0.
df$default = as.integer(ifelse(df$default == "Yes", 1, 0))
df$student = as.integer(ifelse(df$student == "Yes", 1, 0))

#We create an empty dataframe to store the results.
df_results = data.frame(matrix(ncol=3,nrow=100,
                               dimnames=list(NULL,
                               c("val_set_error",
                                 "false_neg", "def_ratio"))))
# a)
glmfit_a = glm(default ~ income + balance, data = df, family =
                  binomial(link = "logit"))
print(summary(glmfit_a))
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

```r
# b)
#Train-set split (70% train-30% test)
df$id = 1:nrow(df)

train = df %>% dplyr::sample_frac(0.70)
test = dplyr::anti_join(df, train, by = 'id')

# Training the logistic regression model
glmfit_b = glm(default ~ income + balance, data = train, family =
                 binomial(link = "logit"))
print(summary(glmfit_b))
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4686  -0.1370  -0.0531  -0.0189   3.3625
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.169e+01  5.342e-01 -21.882  < 2e-16 ***
## income       1.787e-05  6.072e-06   2.944  0.00324 **
## balance      5.767e-03  2.795e-04  20.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2030.3  on 6999  degrees of freedom
## Residual deviance: 1070.9  on 6997  degrees of freedom
## AIC: 1076.9
##
## Number of Fisher Scoring iterations: 8
```

```r
# Getting the predictions for the test dataset
preds_test = predict(glmfit_b,test, type = "response")

#Since we get the probabilities with the regression model, we used a 0.5
#threshold.
preds = ifelse(preds_test>0.5, 1, 0)

#To compute the validation set error and false positive ratio we use the
#confusion matrix.
temp = table(test$default, preds)
print(temp)
```

```
##    preds
##        0    1
##   0 2887   11
##   1   71   31
```

```r
#Misclassification rate, false negative ratio and default ratio
#calculations are stored in the dataframe.
val_set_error = (temp[2]+temp[3])/sum(temp)
false_neg = temp[2]/sum(temp)
def_ratio = sum(test$default)/sum(temp)

paste("Misclassification rate:",val_set_error)
```

```
## [1] "Misclassification rate: 0.0273333333333333"
```

```r
paste("False negative rate:",false_neg)
```

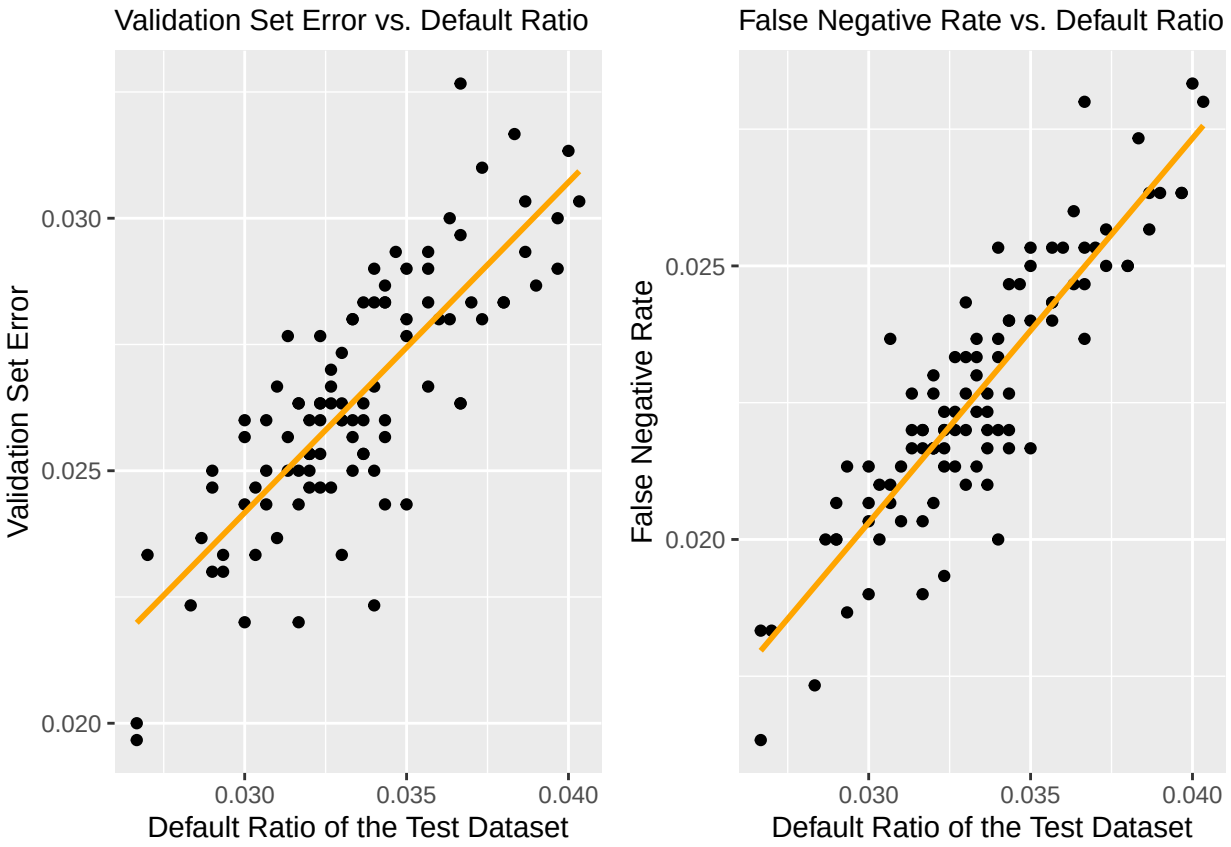
```
## [1] "False negative rate: 0.0236666666666667"
```

```r
paste("Default ratio:",def_ratio)
```

```
## [1] "Default ratio: 0.034"
```

```r
# c)
#We repeat the process 100 times using 100 different splits.
for(i in 1:100) {
  #Train-set split (70% train-30% test)
  df$id = 1:nrow(df)

  train = df %>% dplyr::sample_frac(0.70)
  test = dplyr::anti_join(df, train, by = 'id')

  # Training the logistic regression model
  glmfit = glm(default ~ income + balance, data = train, family =
                 binomial(link = "logit"))

  # Getting the predictions for the test dataset
  preds_test = predict(glmfit,test, type = "response")
```

```
  #Since we get the probabilities with the regression model, we used a 0.5
  #threshold.
  preds = ifelse(preds_test>0.5, 1, 0)

  #To compute the validation set error and false positive ratio we use the
  #confusion matrix.
  temp = table(test$default, preds)

  #Misclassification rate, false negative ratio and default ratio
  #calculations are stored in the dataframe.
  df_results$val_set_error[i] = (temp[2]+temp[3])/sum(temp)
  df_results$false_neg[i] = temp[2]/sum(temp)
  df_results$def_ratio[i] = sum(test$default)/sum(temp)
}

# Creating the plots
val_set_error_plot = ggplot(df_results, aes(x=def_ratio, y=val_set_error)) +
  geom_point() +
  geom_smooth(method=lm , color="orange", se=FALSE) +
  labs(title = "Validation Set Error vs. Default Ratio",
       x = "Default Ratio of the Test Dataset",
       y = "Validation Set Error") +
  theme(plot.title = element_text(size=11))

false_neg_plot = ggplot(df_results, aes(x = def_ratio, y = false_neg)) +
  geom_point() +
  geom_smooth(method=lm , color="orange", se=FALSE) +
  labs(title = "False Negative Rate vs. Default Ratio",
       x = "Default Ratio of the Test Dataset",
       y = "False Negative Rate") +
  theme(plot.title = element_text(size=11))

grid.arrange(val_set_error_plot, false_neg_plot, nrow = 1)
```



We observe a positive relationship in both graphs stating that as the default ratio in the test dataset increases, misclassification error and the false negative rate also increase. One possible explanation for this may be the imbalanced dataset. The algorithm in an imbalanced dataset is biased toward the majority as it has more observations from that class, learning more from the observations. Therefore, in that case the model is more likely to correctly classify the majority but fail to do the same for the minority. Therefore, it's not surprising to see an increase in the false negatives as the default ratio of the test dataset increase.

Since misclassification error includes both false negatives and false positives, we again observe the same positive relationship. However, comparing both graphs we can see that for the the false negative rate vs default ratio graph we observe a stronger positive relaationship as the data points are clustered around the best fit line, on the other graph we observe a weaker positive relationship where the data points are more dispersed from the best fit line. One possible explanation for that can be the effect of the false positives.