Compulsory exercise 1 with short solutions

TMA4268 Statistical Learning V2018

Martina Hall, Martina.Hall@ntnu.no and Mette Langaas

18 May, 2018

Problem 1 - Core concepts in statistical learning

We consider a regression problem, where the true underlying curve is $f(x) = -x + x^2 + x^3$ and we are considering $x \in [-3, 3]$.

This non-linear curve is only observed with added noise (either a random phenomenon, or unobservable variables influence the observations), that is, we observe $y = f(x) + \varepsilon$. In our example the error is sampled from $\varepsilon \sim N(0, 2^2)$.

In real life we are presented with a data set of pairs (x_i, y_i), $i = 1, \ldots, n$, and asked to provide a prediction at a value x. We will use the method of K nearest neighbour regression to do this here.

We have a training set of $n = 61$ observations (x_i, y_i), $i = 1, \ldots, n$. The KNN regression method provides a prediction at a value x by finding the closes K points and calculating the average of the observed y values at these points.

In addition we have a test set of $n = 61$ observations (at the same grid points as for the training set), but now with new observed values y.

We have considered $K = 1, \ldots, 25$ in the KNN method. Our experiment has been repeated $M = 1000$ times (that is, M versions of training and test set).

a) Training and test MSE

In the Figure 2 (above) you see the result of applying the KNN method with $K = 1, 2, 10, 25$ to our training data, repeated for M different training sets (blue lines). The black lines show the true underlying curve.

- Comment briefly on what you see.
- Does a high or low value of K give the most flexible fit?

Answers:

Here $K = 1$ gives predicted values that on average (over the training sets) are not far from the true curve, but the large variability. Increasing the number of neighbours to $K = 2$ gives less variance in the preditions, and still on average are not far from the true curve. Increasing the number of neighbours further to $K = 10$, the variance is even more reduced, but the valued on the edge of our training set (at =-3 and x=3) are not well estimated, and $K = 25$ gives little flexibility to the curve and doesn't fit well, especially on the edges of the data set. Hence, a low value of K gives the most flexible fit.

In Figure 3 (below) you see mean-squared errors (mean of squared differences between observed and fitted values) for the training set and for the test set (right panel for one training and one test set, and left panel for M).

- Comment on what you see.
- What do you think is the "best" choice for K?

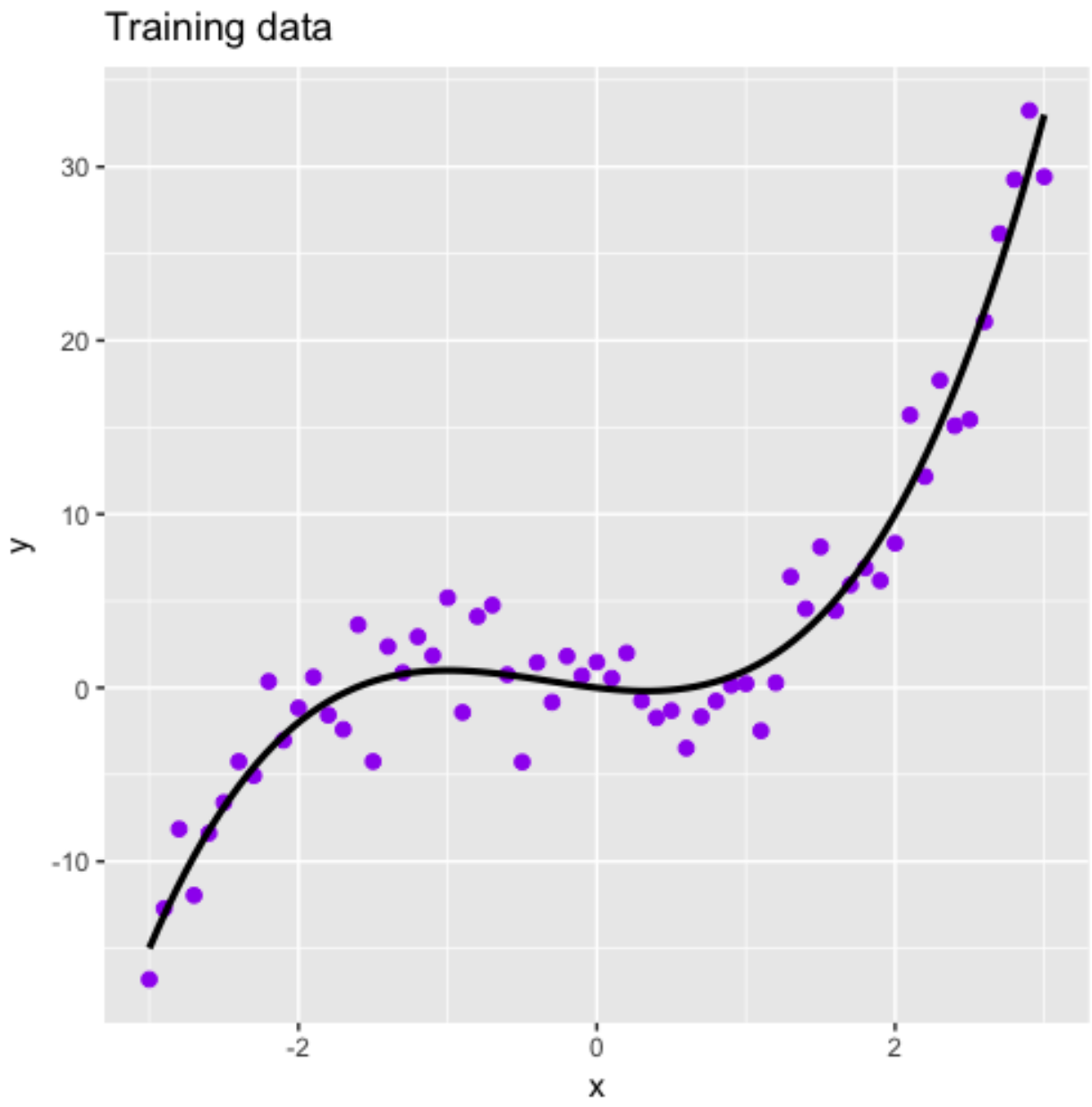


Figure 1: Figure 1

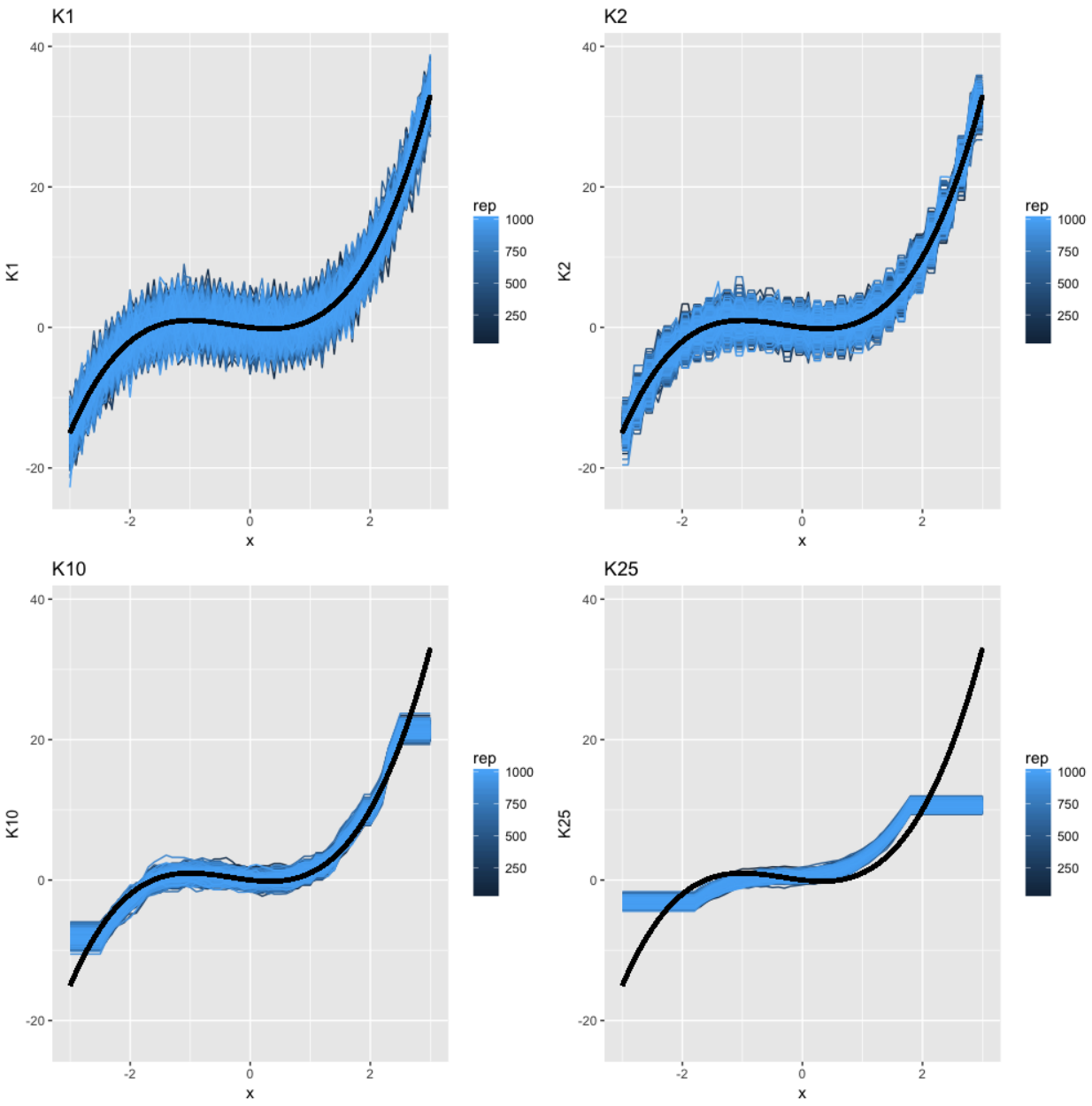


Figure 2: Figure 2

Answers:

We see that larger values of K give larger variations in the MSE for both the test and the train set than for the smaller choices of K. For the training set, the MSE increases linearly with K, i.e. worse fit, but for the test set, there seems to be a small decrease at $K = 5$ before it increases again. Based on this, we would choose $K = 5$ as the best choice for K.

Remark: in real life we do not know the true curve, and need to use the test data to decide on model flexibility (choosing K).

b) Bias-variance trade-off

Now we leave the real world situation, and assume we know the truth (this is to focus on bias-variance trade-off). You will not observe these curves in real life - but the understanding of the bias-variance trade-off is a core skill in this course!

In the Figure 4 (below) you see a plot of estimated squared bias, estimated variance, true irreducible error and the sum of these (labelled total) and averaged over all values of x

The the squared bias and the variance is calculated based on the predicted values and the "true" values (without the added noise) at each x.

- Explain how that is done. Hint: this is what the M repeated training data sets are used for.
- Focus on Figure 4. As the flexibility of the model increases (K decreases), what happens with
  - the squared bias,
  - the variance, and
  - the irreducible error?
- What would you recommend is the optimal value of K? Is this in agreement with what you found in a)?

Answers:

- How this is done? The bias and variance of $f(\hat{x}_0)$ - when the true value is $f(x_0)$ is defined by

$$\text{Bias}(\hat{f}(x_0)) = \text{E}[\hat{f}(x_0) - f(x_0)]$$

$$\text{Var}(\hat{f}(x_0)) = \text{E}[\hat{f}(x_0)^2] - \text{E}[\hat{f}(x_0)]^2$$

  Using the M resamples of the test data, we estimate the mean of $\hat{f}(x_0)$ by the average over the M predicted values at x_0, $\text{Ave}(\hat{f}(x_0))$, and the variance by the emipircal variance $\frac{1}{M-1}\sum_{i=1}^{M}(\hat{f}(x_0) - \text{Ave}(\hat{f}(x_0)))^2$.

$$\text{E}\left[\left(Y - \hat{f}\left(x_0\right)\right)^2\right] = \underbrace{\text{E}(\epsilon)}_{\text{Irredusible error}} + \underbrace{\text{Var}\left(\hat{f}\left(x_0\right)\right)}_{\text{Variance}} + \underbrace{\left[\text{E}\left(\hat{f}\left(x_0\right)\right) - \text{E}\left(Y\right)\right]^2}_{\text{Squared bias}}$$

- From Figure 4, we se that as K increases the squared bias increases rapidly, the variance is slightly reduced, but the irreducible error stays constant. Looking at the total MSE (squared bias plus variance) we see that the optimal K-value lies somewhere between 3 and 5.
- Extra: We observe that the "optimal" K varies for different x_0s. For $x_0 = -2$ the best K is between 5 and 10. For $x_0 = 0$ most values of K is ok, and for $x_0 = 1$ the best is around $K = 10$. For $x_0 = 2.5$ K needs to be below 15. The reason for showing this is to see some of the x_0 values in our interval $[-3, 3]$ that is given equal weight in deciding the optimal K.

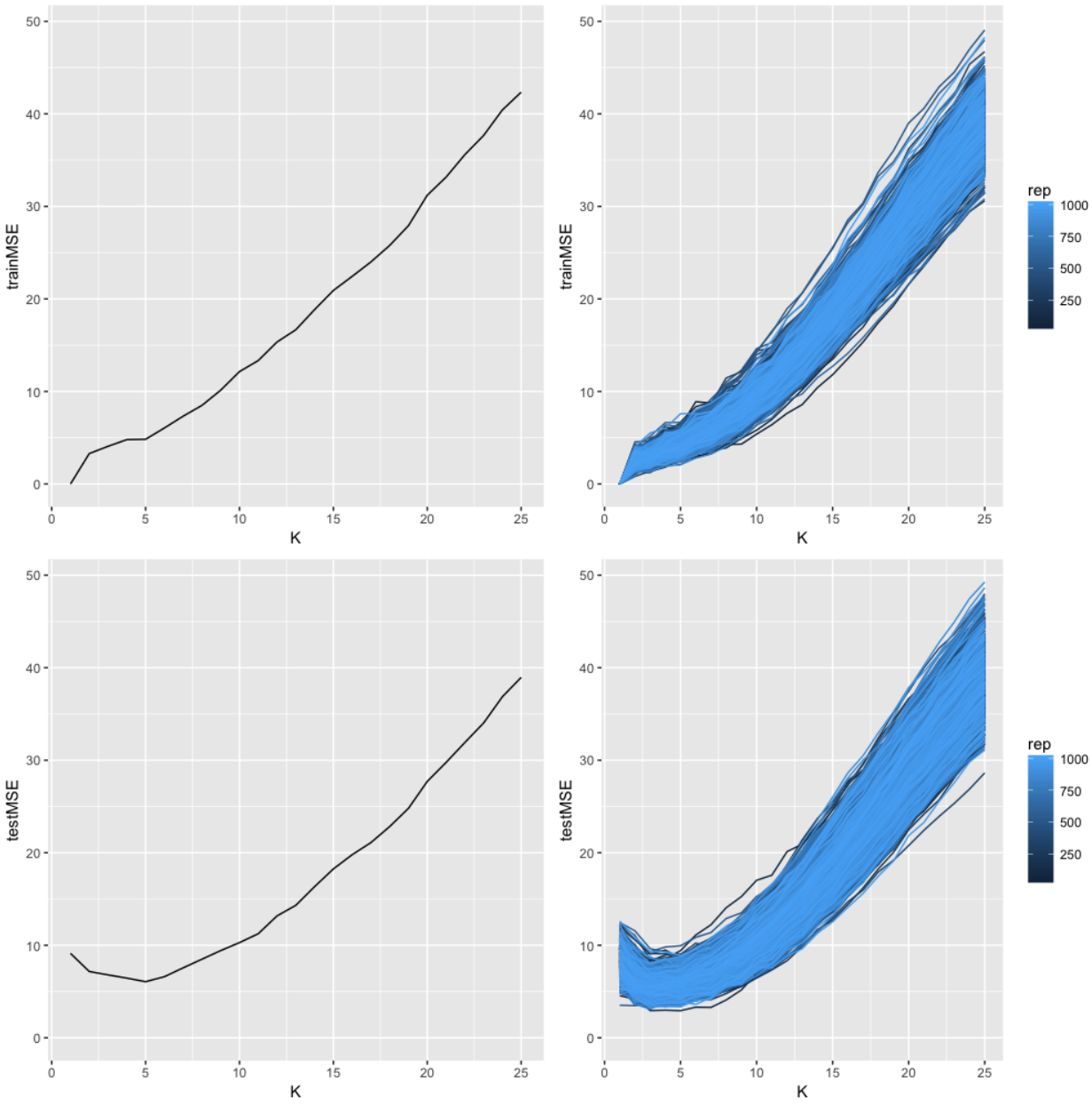


Figure 3: Figure 3

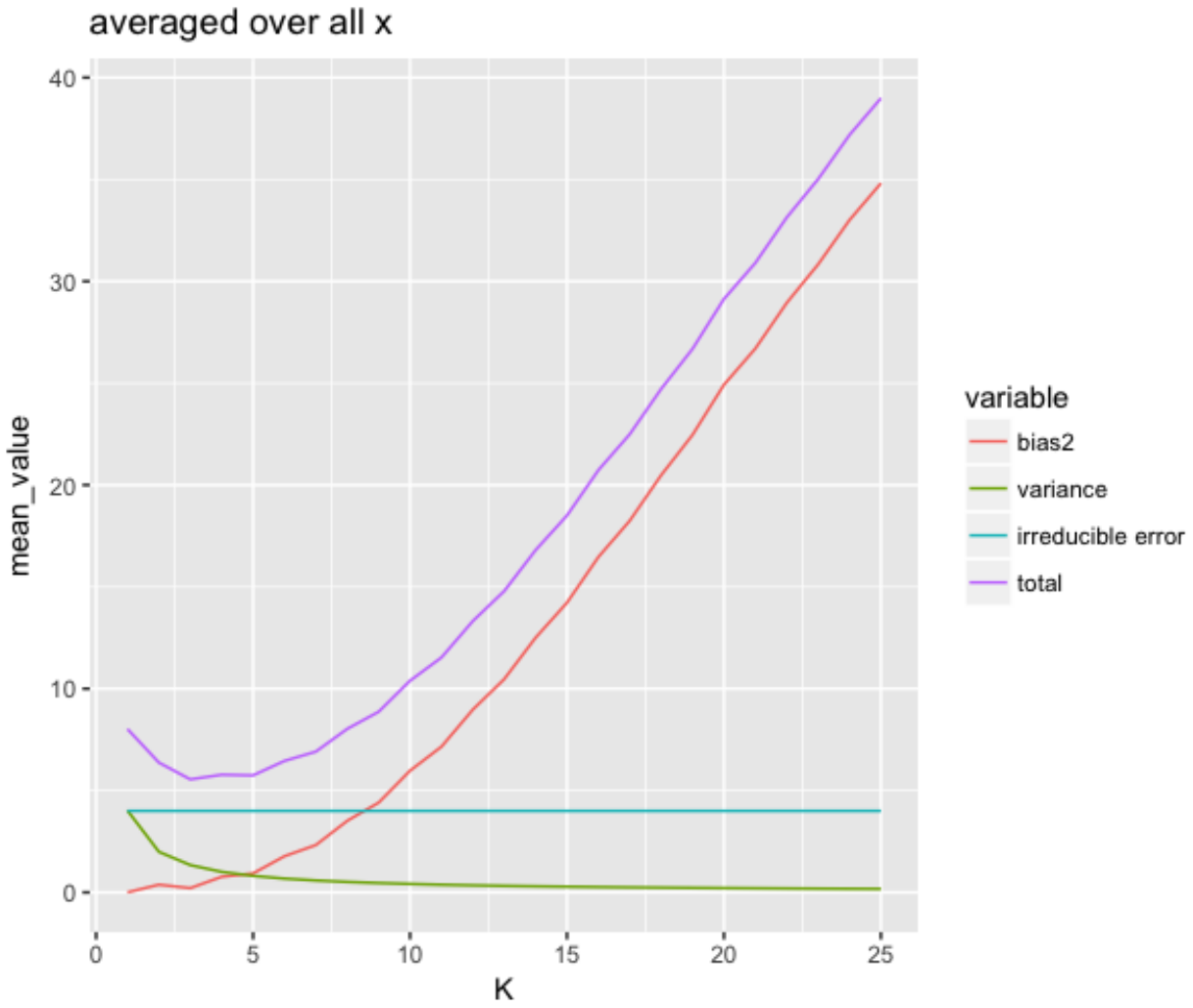


Figure 4: Figure 4

Extra: We have chosen to also plot curves at four values of x - Figure 5 (below). Based on these four curves, that would you recommend is the optimal value of K? Is this in agreement with what you found previously (averaged over x)?

For completeness the R code used is given in the end of this file (listed here with `M=100` but `M=1000` was used). You do not need to run the code, this is just if you have questions about how this was done.

Problem 2 - Linear regression

The Framingham Heart Study is a study of the etiology (i.e. underlying causes) of cardiovascular disease, with participants from the community of Framingham in Massachusetts, USA. For more more information about the Framingham Heart Study visit https://www.framinghamheartstudy.org/. The dataset used in here is subset of a teaching version of the Framingham data, used with permission from the Framingham Heart Study.

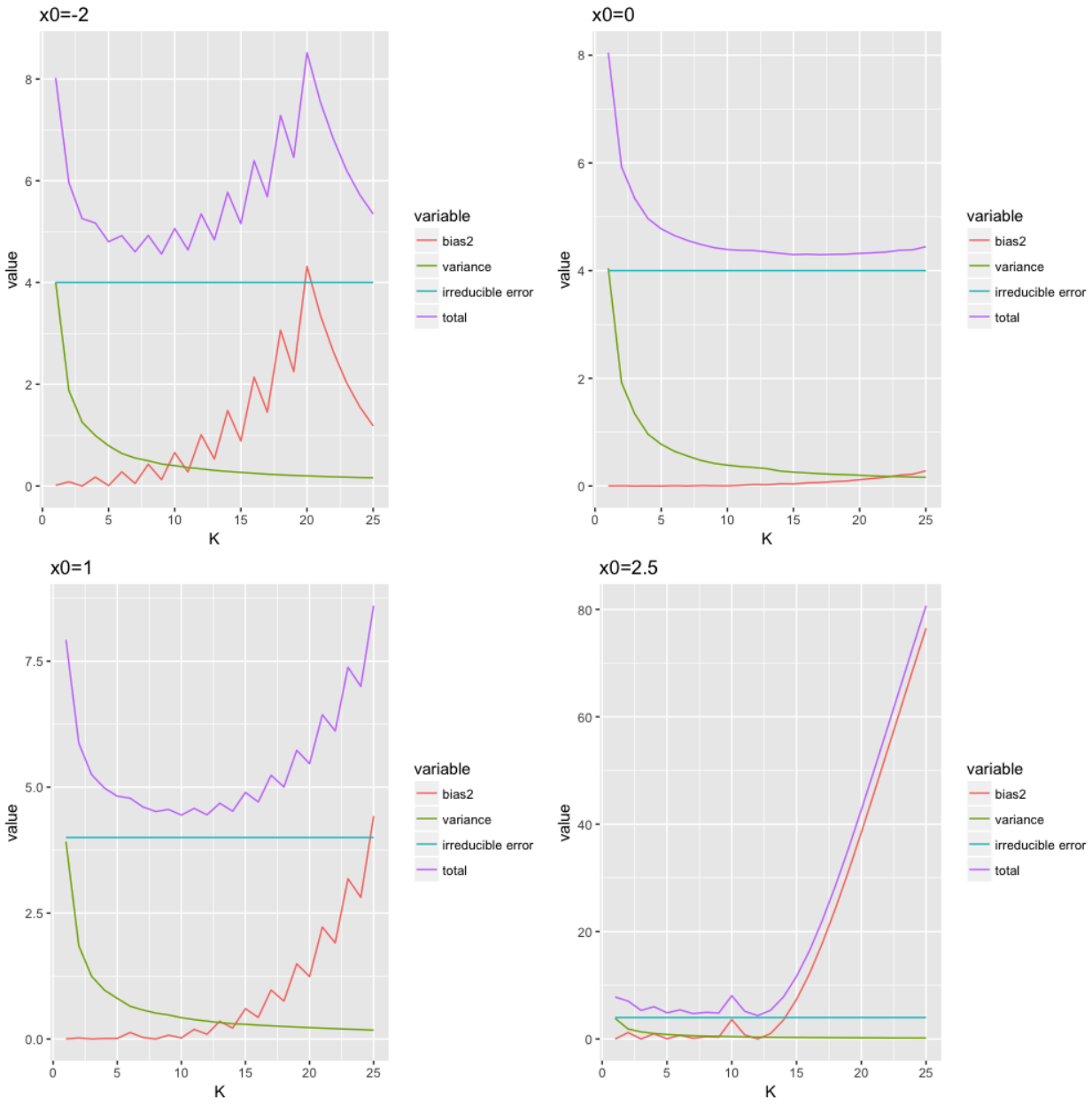


Figure 5: Figure 5

We will focus on modelling systolic blood pressure using data from n = 2600 persons. For each person in the data set we have measurements of the seven variables

- `SYSBP` systolic blood pressure,
- `SEX` 1=male, 2=female,
- `AGE` age in years at examination,
- `CURSMOKE` current cigarette smoking at examination: 0=not current smoker, 1= current smoker,
- `BMI` body mass index,
- `TOTCHOL` serum total cholesterol, and
- `BPMEDS` use of anti-hypertensive medication at examination: 0=not currently using, 1=currently using.

A multiple normal linear regression model was fitted to the data set with `-1/sqrt(SYSBP)` as response and all the other variables as covariates.

```
library(ggplot2)
#data = read.table("https://www.math.ntnu.no/emner/TMA4268/2018v/data/SYSBPreg3uid.txt")
data = read.table("~/WWWemner/TMA4268/2018v/data/SYSBPreg3uid.txt")
dim(data)
colnames(data)
modelA=lm(-1/sqrt(SYSBP) ~ .,data = data)
summary(modelA)
```

```
## [1] 2600    7
## [1] "SYSBP"    "SEX"      "AGE"      "CURSMOKE" "BMI"      "TOTCHOL"
## [7] "BPMEDS"
##
## Call:
## lm(formula = -1/sqrt(SYSBP) ~ ., data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0207366 -0.0039157 -0.0000304  0.0038293  0.0189747
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.103e-01  1.383e-03 -79.745  < 2e-16 ***
## SEX         -2.989e-04  2.390e-04  -1.251 0.211176
## AGE          2.378e-04  1.434e-05  16.586  < 2e-16 ***
## CURSMOKE    -2.504e-04  2.527e-04  -0.991 0.321723
## BMI          3.087e-04  2.955e-05  10.447  < 2e-16 ***
## TOTCHOL      9.288e-06  2.602e-06   3.569 0.000365 ***
## BPMEDS       5.469e-03  3.265e-04  16.748  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005819 on 2593 degrees of freedom
## Multiple R-squared:  0.2494, Adjusted R-squared:  0.2476
## F-statistic: 143.6 on 6 and 2593 DF,  p-value: < 2.2e-16
```

a) Understanding model output

We name the model fitted above `modelA`.

- Write down the equation for the fitted `modelA`.
- Explain (with words and formula) what the following in the `summary`-output means.

- `Estimate` - in particular interpretation of `Intercept`
- `Std.Error`
- `t value`
- `Pr(>|t|)`
- `Residual standard error`
- `F-statistic`

Answers:

- Model A:

  $$-1/\sqrt{\text{SYSBP}} = \beta_0 + \beta_1\text{SEX} + \beta_2\text{AGE} + \beta_3\text{CURSMOKE} + \beta_4\text{BMI} + \beta_5\text{TOTCHOL} + \beta_6\text{BPMEDS} + \epsilon$$

  with the fitted version

  $$\widehat{1/\sqrt{\text{SYSBP}}} = -0.110 - 0.0003\text{SEX} + 0.0002\text{AGE} - 0.0003\text{CURSMOKE} + 0.0003\text{BMI} + 0.00001\text{TOTCHOL} + 0.0055\text{BPMEI}$$

- The `Estimate` is the estimated regression coefficients, and are given by $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$. The interpretation of $\hat{\beta}_j$ is that when all other covariates are kept constant and the covariate x_j is increased to from x_j to $x_j + 1$ then the response increases by $\hat{\beta}_j$. Example, holding all other variables constant, an increase of BMI from 25 to 26 will increase the response $-1/\sqrt{\text{SYSBP}}$ by 0.00031. Similarily, for the binary variables, the coefficient estimates represents the change in the response when changing levels of the variable with one unit. For a female, the response will hence be reduced by 0.0003 compared to a male (with the same values of all the other covariate). For all variables, negative value of the estimates give reduced response when increasing the corresponding variable, while positive estimates give increased response when increasing the corresponding variable. The intercept, β_0 can be found by setting all other coefficients to zero. This involves also setting the covariate SEX to 0 - which has no meaning since SEX is coded as 1 for male and 2 for female.
- The `Std.Error` $\hat{SD}(\hat{\beta}_j)$ of the estimated coefficients is given by the square root of the diagonal entries of $(\mathbf{X}^T\mathbf{X})^{-1}\hat{\sigma}^2$, where $\hat{\sigma} = \text{RSS}/(n-p-1)$. Here $n = 2600$ and $p = 6$.
- The `t value` is the t-statistic $t = \frac{\hat{\beta}_j - \beta_j}{\hat{SD}(\hat{\beta}_j)}$, when we assume that $\beta_j = 0$.
- The `Pr(>|t|)` is the two-sided p-value for the null hypothesis $\beta_j = 0$. The p-value is calculated as the probability of observing a test staistics equal to $|t|$ or larger in absolute value, assuming that the null hypothesis is true. A p-value less than 0.05 is considered statistically significant at a 5% significance level.
- The residual standard error is the estimate of the standard deviation of ϵ, and is given by $\text{RSS}/(n-p-1)$ where $\text{RSS}=\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- The `F-statistic` is used test the hypothesis that all regression coefficients are zero,

  $$\begin{aligned} H_0 &: \beta_1 = \beta_2 = \cdots = \beta_p = 0 \quad \text{vs} \\ H_1 &: \text{at least one } \beta \text{ is} \neq 0 \end{aligned}$$

  and is computed by

  $$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

where $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$, $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, n is the number of observations and p is the number of covariates (and $p+1$ the number of estimated regression parameters). If the p-value is less than 0.05, we reject the hypothesis that there are no coefficients with effect on the outcome in the model.

b) Model fit

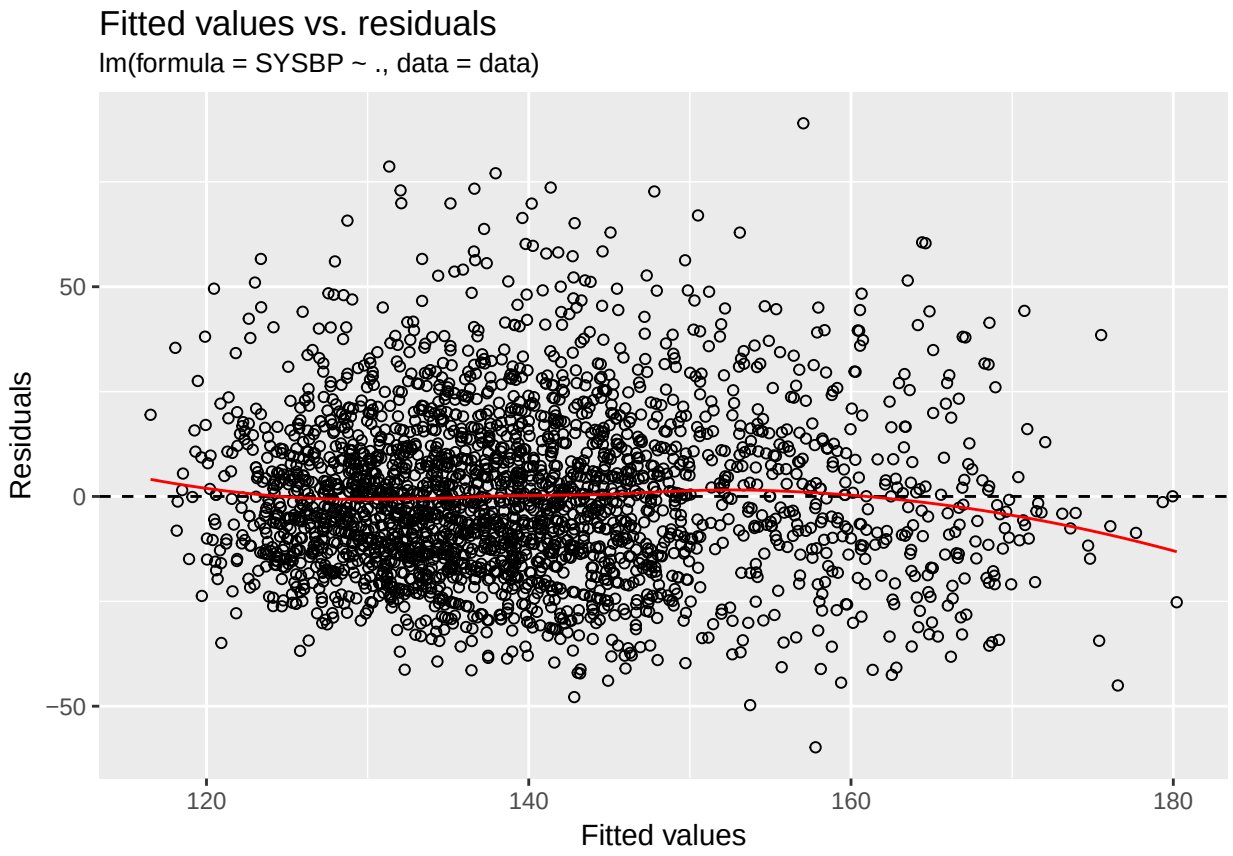
- What is the proportion of variability explained by the fitted `modelA`? Comment.
- Use diagnostic plots of "fitted values vs. standardized residuals"" and "QQ-plot of standardized residuals" (see code below) to assess the model fit.
- Now fit a model, call this `modelB`, with `SYSBP` as response, and the same covariates as for `modelA`. Would you prefer to use `modelA` or `modelB` when the aim is to make inference about the systolic blood pressure?

```
# residuls vs fitted
ggplot(modelA, aes(.fitted, .resid)) + geom_point(pch = 21) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(se = FALSE, col = "red", size = 0.5, method = "loess") +
  labs(x = "Fitted values", y = "Residuals", title = "Fitted values vs. residuals", subtitle = deparse(m

# qq-plot of residuals
ggplot(modelA, aes(sample = .stdresid)) +
  stat_qq(pch = 19) +
  geom_abline(intercept = 0, slope = 1, linetype = "dotted") +
  labs(x = "Theoretical quantiles", y = "Standardized residuals", title = "Normal Q-Q", subtitle = depar

# normality test
library(nortest)
ad.test(rstudent(modelA))
```

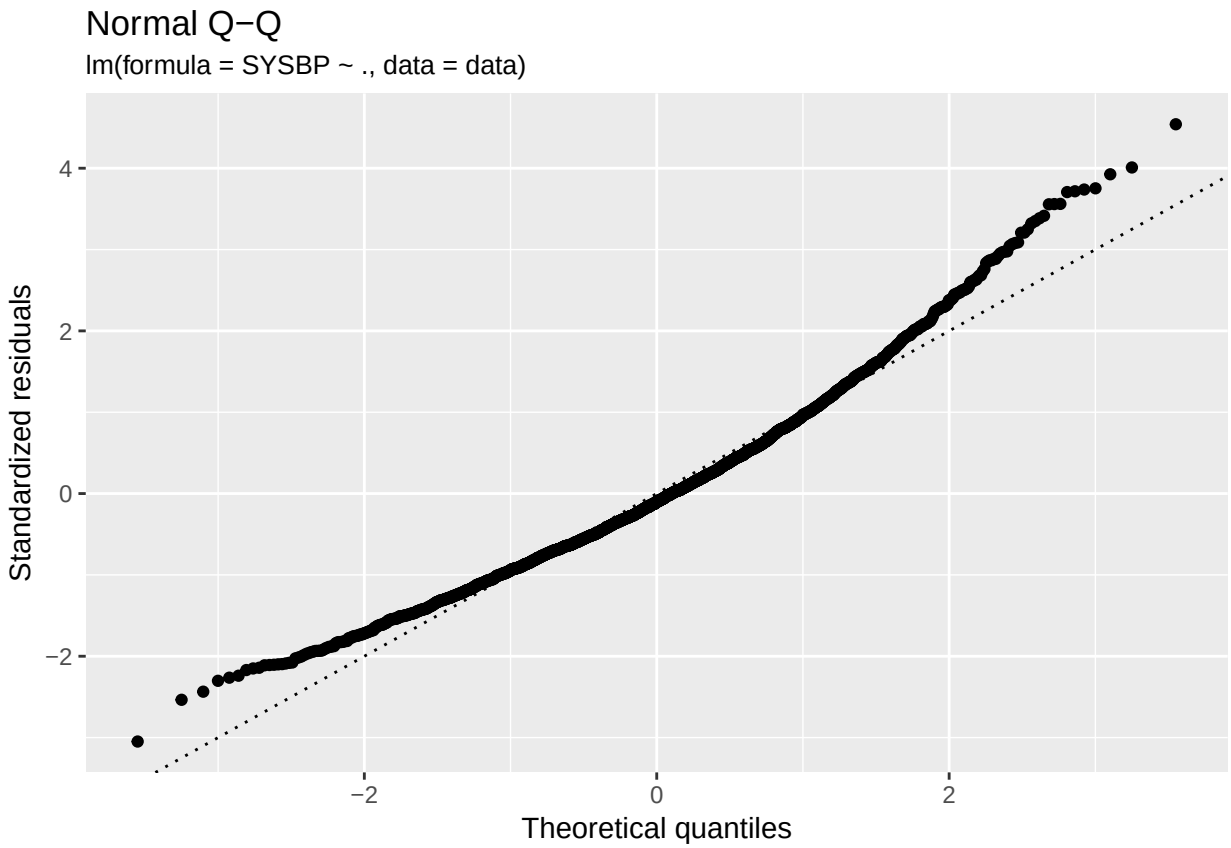
Answers:

- The R^2 statistic gives the proportion of variance explained by the model. In this model, the proportion of variability in $Y = -1/\sqrt{\text{SYSBP}}$ explained by the data X is 0.2494. Since the range of R^2 is from 0 to 1, where for 1 all the variance in the response is explained by the regression model, we observe a fairly low number and we would have prefered it higher. However, these are medical data with low signal-to-noise ratio.
- Looking at the diagnostic plots, the model fit looks good. The fitted values vs residuals plot is nice with semingly random spread and the QQ-plot looks nice as the plotted values follows the normal line. In addition, the Anderson-Darling normality test does not reject the hypothesis of normality.
- For model B, we no longer model $-1/\sqrt{\text{SYSBP}}$, but rather SYSBP. This makes interpreation easier. However, looking at the diagnostic plots, we see that the QQ-plot looks suspicious at the tails, and the Anderson-Darling test rejects the null hypothesis of normal distribution.

```
modelB = lm(SYSBP ~ .,data = data)
summary(modelB)

library(ggplot2)
# residuls vs fitted
ggplot(modelB, aes(.fitted, .resid)) + geom_point(pch = 21) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(se = FALSE, col = "red", size = 0.5, method = "loess") +
  labs(x = "Fitted values", y = "Residuals", title = "Fitted values vs. residuals", subtitle = deparse(m
```




```
# qq-plot of residuals
ggplot(modelB, aes(sample = .stdresid)) +
  stat_qq(pch = 19) +
  geom_abline(intercept = 0, slope = 1, linetype = "dotted") +
  labs(x = "Theoretical quantiles", y = "Standardized residuals", title = "Normal Q-Q", subtitle = depa
```




```
# normality test
library(nortest)
ad.test(rstudent(modelB))
```

```
##
## Call:
## lm(formula = SYSBP ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.800 -13.471  -1.982  11.063  88.959
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 56.505170   4.668798  12.103  < 2e-16 ***
## SEX         -0.429973   0.807048  -0.533  0.59424
## AGE          0.795810   0.048413  16.438  < 2e-16 ***
## CURSMOKE    -0.518742   0.853190  -0.608  0.54324
## BMI          1.010550   0.099770  10.129  < 2e-16 ***
## TOTCHOL      0.028786   0.008787   3.276  0.00107 **
## BPMEDS      19.203706   1.102547  17.418  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.65 on 2593 degrees of freedom
## Multiple R-squared:  0.2508, Adjusted R-squared:  0.249
## F-statistic: 144.6 on 6 and 2593 DF,  p-value: < 2.2e-16
```

```
##
##
##  Anderson-Darling normality test
##
## data:  rstudent(modelB)
## A = 13.2, p-value < 2.2e-16
```

c) Confidence interval and hypothesis test

We use `modelA` and focus on addressing the association between BMI and the response.

- What is the estimate $\hat{\beta}_{\text{BMI}}$ (numerically)?
- Explain how to interpret the estimated coefficient $\hat{\beta}_{\text{BMI}}$.
- Construct a 99% confidence interval for β_{BMI} (write out the formula and calculate the interval numerically). Explain what this interval tells you.
- From this confidence interval, is it possible for you know anything about the value of the p-value for the test $H_0 : \beta_{\text{BMI}} = 0$ vs. $H_1 : \beta_{\text{BMI}} \neq 0$? Explain.

Answers:

- $\hat{\beta} = (X^T X)^{-1} X^T Y$. From the summary output we find that $\hat{\beta}_{\text{BMI}} = 0.0003$. This is the average increase in $-1/sqrt(SYSBP)$ for a unit increase in BMI. Hence, keeping all other covariates fixed - having a BMI of 24 instead of 23, the value of $-1/sqrt(SYSBP)$ will on average increase with 0.0003.
- For linear regression where the distribution of the estimated coefficients are assumed to follow a t-distribution, we have that the $(1-\alpha)100\%$-confidence interval is given by

  $$\hat{\beta} \pm t_{\alpha/2,df} SD(\hat{\beta})$$

  For $\hat{\beta}_{BMI}$ the 99% confidence interval is hence given by

  $$[\hat{\beta}_{BMI} - t_{0.005,n-p-1} SD(\hat{\beta}_{BMI}), \hat{\beta}_{BMI} + t_{0.005,n-p-1} SD(\hat{\beta}_{BMI})]$$

  This means that before we have collected the data this interval has a 99% chance of covering the true value of β_{BMI}. After the interval is made - now this is [0.00023, 0.00038] the the true value is either within the interval or not. But, colleting new data and making 99% CIs, then 99% of these will on average cover the true β_{BMI}.
- Since the interval does not cover 0, we know that the p-value is less than 0.01.

```
n = dim(data)[1]
p = dim(data)[2]-1
betahat=modelA$coefficients[5]
sdbetahat=summary(modelA)$coeff[5,2]
UCI = betahat + qt(0.005, df = n-p-1, lower.tail = F)*sdbetahat
LCI = betahat - qt(0.005, df = n-p-1, lower.tail = F)*sdbetahat
c(LCI, UCI)
```

```
##          BMI          BMI
## 0.0002325459 0.0003848866
```

d) Prediction

Consider a 56 year old man who is smoking. He is 1.75 meters tall and his weight is 89 kilograms. His serum total cholesterol is 200 mg/dl and he is not using anti-hypertensive medication.

```
names(data)
new=data.frame(SEX=1,AGE=56,CURSMOKE=1,BMI=89/1.75^2,TOTCHOL=200,BPMEDS=0)
```

```
## [1] "SYSBP"    "SEX"      "AGE"      "CURSMOKE" "BMI"      "TOTCHOL"
## [7] "BPMEDS"
```

- What is your best guess for his `-1/sqrt(SYSBP)`? To get a best guess for his `SYSBP` you may take the inverse function of `-1/sqrt` (this would be a first order Taylor expansion).
- Construct a 90% prediction interval for his systolic blood pressure `SYSBP`. Comment. Hint: first contruct values on the scale of the response `-1/sqrt(SYSBP)` and then transform the upper and lower limits of the prediction interval.
- Do you find this prediction interval useful? Comment.

Answers:

Find best guess by prediction, and 90% prediction interval.

```
pred = predict(modelA, newdata = new)
pred
f.inv = function(x) 1/x^2
sys = f.inv(pred)
#pred. interval
f.ci = predict(modelA, newdata = new, level = 0.9, interval = "prediction")
f.ci
sys.ci = f.inv(f.ci)
sys.ci
```

```
##           1
## -0.08667246
##           fit         lwr         upr
## 1 -0.08667246 -0.09625664 -0.07708829
##        fit      lwr      upr
## 1 133.1183 107.9291 168.2764
```

This prediction interval is very large and doesn't really tell us much. A person with our characteristics on average has a 90% chance of having a systolic blood pressure between 108 and 168, and looking at the table given in http://www.heart.org/HEARTORG/Conditions/HighBloodPressure/KnowYourNumbers/Understanding-Blood-Pressure-Readings_UCM_301764_Article.jsp#.WnLqWOYo_AI, we sse that this interval covers almost all the levels from normal to high blood pressure. It seems our model is better for inference than prediction.

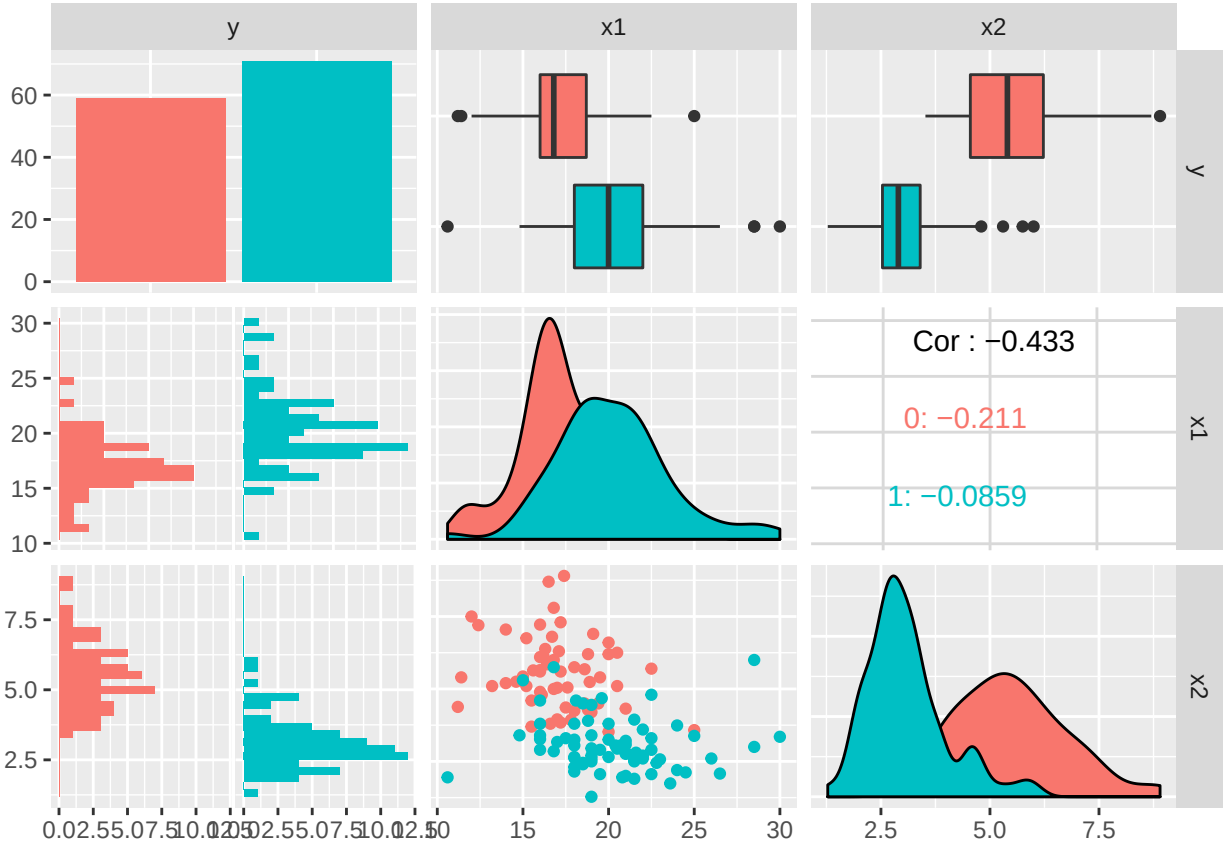
Problem 3 - Classification

In this problem, we use a `wine` dataset of chemical measurement of two variables, `Color_intensity` and `Alcalinity_of_ash`, on 130 wines from two cultivars in a region in Italy.

The data set is a subset of a data set from https://archive.ics.uci.edu/ml/datasets/Wine, see that page for information of the source of the data.

Below you find code to read the data, plot the data and to divide the data into a training set and a test set. To get your own unique division please change the seed (where it says `set.seed(4268)` you change 4268 to your favorite number).

```
library(ggplot2)
library(GGally)
library(class)
library(MASS)
library(pROC)
#wine=read.csv("https://www.math.ntnu.no/emner/TMA4268/2018v/data/Comp1Wine.csv",sep=" ")
wine = read.csv("~/WWWemner/TMA4268/2018v/data/Comp1Wine.csv",sep=" ")
wine$class=as.factor(wine$class-1)
colnames(wine)=c("y","x1","x2")
ggpairs(wine, ggplot2::aes(color=y))
```




```
n=dim(wine)[1]
set.seed(4268) #to get the same order if you rerun - but you change this to your favorite number
ord = sample(1:n) #shuffle
test = wine[ord[1:(n/2)],]
train = wine[ord[((n/2)+1):n],]
```

In our data the two classes are named `y` and coded $Y = 0$ and $Y = 1$, and we name x_1=`Alcalinity_of_ash`=`x1` and x_2=`Color_intensity`=`x2`.

a) Logistic regression

We assume a logistic regression model for observation i, $i = 1, \ldots, n$:

$$\Pr(Y_i = 1|\mathbf{X} = \mathbf{x}_i) = p_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}}}{1 + e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}}}$$

- Use this expression to show that $\text{logit}(p_i) = \log(\frac{p_i}{1-p_i})$ is a linear function.
- Fit a logistic regression model on `y~x1+x2` to the training set.
- Give an interpretation of $\hat{\beta}_1$ and $\hat{\beta}_2$.
- We use the rule to classify to class 1 for an observation with covariates $\mathbf{x}$ if $\hat{\Pr}(Y = 1 \mid \mathbf{x}) > 0.5$. Write down the formula for the class boundary between the classes. What type of boundary is this?
- Make a plot with the training observations and the class boundary. Hint: in `ggplot` points are added with `geom_point` and a line with `geom_abline(slope=b, intercept=a)` where a and b comes from your class boundary, and title with `ggtitle`.
- Use the `summary` output to manually derive the predicted probability $\hat{\Pr}(Y = 1 \mid x_1 = 17, x_2 = 3)$. What is the interpretation of this value?
- Compute predicted probabilites for all observations in the test set.
- Make the confusion table for the test set when using 0.5 as cutoff for the probabilities. Calculate the sensitivity and specificity on the test set. How would you evaluate the performance of this classification?

Answers:

- $$\log\left(\frac{p_i}{1-p_i}\right) = \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}}{1-\frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}}\right) = \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}}{\frac{1}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}}\right)$$
$$= \log(e^{\beta_0+\beta_1 x_1+\beta_2 x_2}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

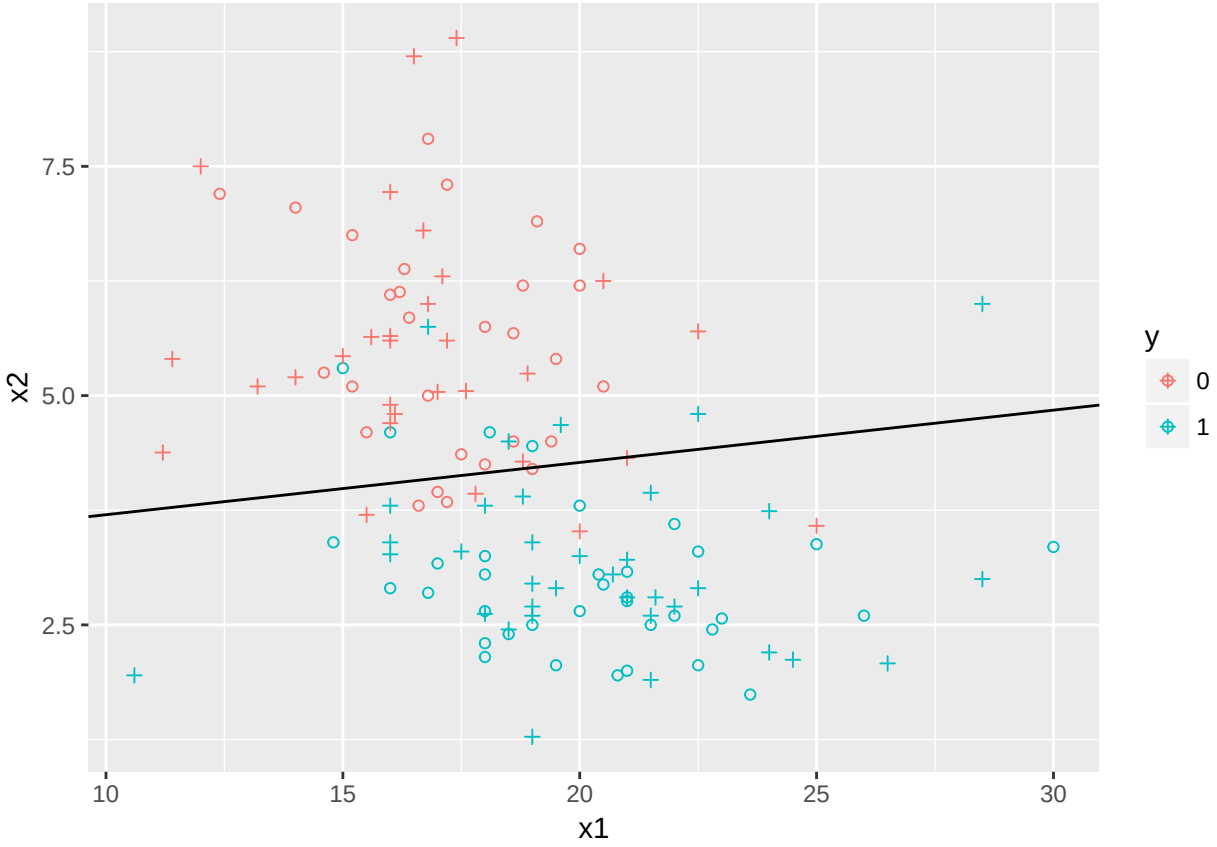
```
fit = glm(y~x1+x2, data = train, family = "binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.46217  -0.17536   0.09309   0.28590   2.49572
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   7.3143     5.3382   1.370 0.170626
## x1            0.1332     0.2194   0.607 0.543800
## x2           -2.3361     0.6472  -3.609 0.000307 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89.354  on 64  degrees of freedom
## Residual deviance: 30.027  on 62  degrees of freedom
## AIC: 36.027
```

```
##
## Number of Fisher Scoring iterations: 6
```

- $\hat{\beta}_1$ is the estimated coefficient for `x1` with a value 0.1332. This means that the odds of begin a wine of class 2 is multiplied by $\exp(\hat{\beta}_1) = 1.14$ if `x1` increases by one unit. Further, $\hat{\beta}_2$ is the estimated coefficient for `x2` with value -2.34, which is interpreted as the odds of begin a wine of class 2 is multiplied by $\exp(\hat{\beta}_2) = 0.097$ if `x2` increases by one unit.
- The class boundary is where there is equal probability of both classes, $p_i = 0.5$. Using the logit functon, we find that $\frac{p_i}{1-p_i} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$. With $p_i = 0.5$, the class boundary is given by $1 = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$ and hence $0 = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, and $x_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} x_1$. The class boundary is a straight line and can easily be added to the scatter plot below.

```
betas = fit$coefficients
slope = -betas[2]/betas[3]
intercept = -betas[1]/betas[3]

g1 = ggplot(data=train,aes(x=x1, y=x2, colour=y)) + geom_point(pch = 1)
g1 + geom_point(data = test, pch = 3) + geom_abline(slope=slope,intercept=intercept)
```




```
x=c(1,17,3)
p = exp(fit$coefficients%*%x)/(1+exp(fit$coefficients%*%x))
p
```

```
##           [,1]
## [1,] 0.9289381
```

For a wine with `x1` of 17 and `x2` of 3, there is a 93% probability that the wine is of class 1.

```
pred = predict.glm(fit, newdata = test, type = "response")
predglm = predict.glm(fit, newdata = test, type = "response")
testclass=ifelse(predglm > 0.5, 1, 0)
t = table(test$y, testclass)
t

n = length(test$y)
error = (n-sum(diag(t)))/n
error
library(caret)
```

```
## Loading required package: lattice
```

```
confusionMatrix(as.factor(testclass),reference=as.factor(test$y),positive="1")
```

```
##    testclass
##      0  1
##   0 25  5
##   1  5 30
## [1] 0.1538462
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25  5
##          1  5 30
##
##                Accuracy : 0.8462
##                  95% CI : (0.7352, 0.9237)
##     No Information Rate : 0.5385
##     P-Value [Acc > NIR] : 1.61e-07
##
##                   Kappa : 0.6905
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8571
##             Specificity : 0.8333
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5385
##          Detection Rate : 0.4615
##    Detection Prevalence : 0.5385
##       Balanced Accuracy : 0.8452
##
##        'Positive' Class : 1
##
```

- 10 misclassifications equally divided over both wines, and a error rate pf 0.15. I would say that this classification works pretty well. The sensitivity is $30/35 = 0.86$ (true class 1), and the specificity is $25/30 = 0.83$ (true class 0). This is given that we define class 1 as the "positive class" - if we do differently the sensitivity and specificity will be swapped. For a disease situation it is "easy" to choose the disease as coded as the "postive", but for our wine example - we may use either 0 or 1 as positive. So, if you got sensitivity and specificity swapped - that is ok.

b) K-nearest neighbor classifier

To decide the class of an new observation, the KNN classifier uses the nearest neighbours in the following way,

$$P(Y = 1|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_\prime} I(y_i = j).$$

- Explain this expression does, and what the different elements are.
- Use KNN with $K = 3$ to classify the wines in the test set.
- Make the confusion table for the test set when using 0.5 as cutoff. Calculate the sensitivity and specificity on the test set. How would you evaluate the performance of this classification?
- Repeat with $K = 9$. Which of these two choices of K would you prefer and why? Why don't we just choose K as high or as low as possible?

Answers:

- Given an integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by $\mathcal{N}_\prime$. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_\prime$ whose response values equal j.

```
KNN3 = knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = F)
t3 = table(test$y, KNN3)
t3
apply(t3,1,sum)
n = length(test$y)
error = (n-sum(diag(t3)))/n
error
KNN3probwinning = attributes(knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = TRUE)
KNN3prob <- ifelse(KNN3 == "0", 1-KNN3probwinning, KNN3probwinning)
#cbind(KNN3prob,KNN3,KNN3probwinning) to check that this is correct
KNN3roc=roc(response=test$y,predictor=KNN3prob)
cbind(KNN3roc$threshold, KNN3roc$sens, KNN3roc$specificities)
```

```
##    KNN3
##      0  1
##   0 25  5
##   1  3 32
##  0  1
## 30 35
## [1] 0.1230769
##           [,1]      [,2]      [,3]
## [1,]      -Inf 1.0000000 0.0000000
## [2,] 0.1666667 0.9714286 0.4666667
## [3,] 0.5000000 0.9142857 0.8333333
## [4,] 0.8333333 0.6857143 0.9333333
## [5,]       Inf 0.0000000 1.0000000
```

- KNN3: This gives a rather good result with a misclassification rate of 0.12. Since we have only three neighbours to base the classification on the only possible values for the probability of class 2 is 0, 0.33, 0.67 and 1. For a cut-off of 0.5 the sensitivity is 32/35=0.91 and the specificity is 25/30=0.83.

```
KNN9= knn(train = train[,-1], test = test[,-1], k = 9, cl = train$y, prob = F)
t9 = table(test$y, KNN9)
t9
error = (n-sum(diag(t9)))/n
```

```
error
KNN9probwinning = attributes(knn(train = train[,-1], test = test[,-1], k = 9, cl = train$y, prob = TRUE
KNN9prob <- ifelse(KNN9 == "0", 1-KNN9probwinning, KNN9probwinning)
KNN9roc=roc(response=test$y,predictor=KNN9prob)
cbind(KNN9roc$threshold, KNN9roc$sens, KNN9roc$specificities)
table(KNN3,KNN9)
```

```
##    KNN9
##      0  1
##   0 25  5
##   1  5 30
## [1] 0.1538462
##              [,1]      [,2]      [,3]
##  [1,]       -Inf 1.0000000 0.0000000
##  [2,] 0.05555556 1.0000000 0.1333333
##  [3,] 0.16666667 0.9714286 0.3333333
##  [4,] 0.27777778 0.9428571 0.6666667
##  [5,] 0.38888889 0.8857143 0.7666667
##  [6,] 0.50000000 0.8571429 0.8333333
##  [7,] 0.61111111 0.7142857 0.9000000
##  [8,] 0.72222222 0.6857143 0.9000000
##  [9,] 0.83333333 0.6857143 0.9666667
## [10,] 0.94444444 0.6000000 0.9666667
## [11,]        Inf 0.0000000 1.0000000
##     KNN9
## KNN3  0  1
##    0 27  1
##    1  3 34
```

- KNN9 For $K = 9$ the misclassification rate when using 0.5 as cut-off is 0.15, so higher than for $K = 3$, but similar to logistic regression. The sensitivity is The classifications made by $K = 3$ and $K = 9$ are not that different, 27+34=61 common classifications. But, since $K = 3$ have the lowest misclassification rate we might prefer that, however for $K = 3$ we may have strange boundary effects.
- If we choose $K = 1$ that might lead to a too flexible class boundary, and with $K = n$ this might be too inflexible.

c) LDA (& QDA)

In linear discriminant analysis, with K classes, we assign a class to a new observation based on the posterior probability

$$\Pr(Y = k|\mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^{K} \pi_l f_l(\mathbf{x})},$$

where

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_k})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_k})}.$$

- Explain what is π_k, $\boldsymbol{\mu}_k$, $\mathbf{\Sigma}$ and $f_k(x)$ in our `wine` problem.
- How can we estimate π_k, $\boldsymbol{\mu}_k$ and $\mathbf{\Sigma}$? Compute estimates for these quantities based on the training set.

In a two class problem ($K = 2$) the decision boundary for LDA between class 0 and class 1 is where x satisfies

$$\Pr(Y = 0|\mathbf{X} = \mathbf{x}) = \Pr(Y = 1|\mathbf{X} = \mathbf{x}).$$

- Show that we can express this as

$$\delta_0(\mathbf{x}) = \delta_1(\mathbf{x}), \tag{1}$$

where

$$\delta_k(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log\pi_k; \quad k \in \{0,1\}. \tag{2}$$

- Perform LDA on the training data (using R).
- We use the rule to classify to class 1 for an observation with covariates $\mathbf{x}$ if $\hat{\Pr}(Y = 1 \mid \mathbf{x}) > 0.5$. Write down the formula for the class boundary between the classes.
- Make a plot with the training observations and the class boundary. Add the test observations to the plot (different markings). Hint: in `ggplot` points are added with `geom_points` and a line with `geom_abline(slope=b, intercept=a)` where a and b comes from your class boundary.
- Make the confusion table for the test set when using 0.5 as cut-off. Calculate the sensitivity and specificity on the test set. How would you evaluate the performance of this classification?
- If you where to perform QDA instead of LDA, what would be the most important difference between the QDA and LDA philosophy?

Answers:

- Here π_k is the prior probability that a randomly chosen observation comes from the kth class. We assume that the observations of class k comes from a multivariate normal distribution $f_k(x)$, where $\boldsymbol{\mu}_k$ is the mean of the kth class and Σ is the variance.
- We estimate these in the following way:

$$\begin{aligned}
\hat{\pi}_k &= n_k/n \\
\hat{\mu}_k &= \frac{1}{n_k}\sum_{i:y_i=k} x_i \\
\hat{\boldsymbol{\Sigma}}_k &= \frac{1}{n_k-1}\sum_{i:y_i=k}(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)^T \\
\hat{\boldsymbol{\Sigma}} &= \sum_{k=1}^{K}\frac{n_k-1}{n-K}\cdot\hat{\boldsymbol{\Sigma}}_k.
\end{aligned}$$

For the training dataset we have that

```
n=dim(train)[1]
train0 = train[which(train$y==0),2:3]
train1 = train[which(train$y==1),2:3]
print("pi0 and pi1")
pi0=dim(train0)[1]/n; pi1=dim(train1)[1]/n
c(pi0,pi1)
print("mu")
mu0=apply(train0,2,mean)
mu1=apply(train1,2,mean)
mu0
mu1
print("Sigma")
Sigma=((dim(train0)[1]-1)*var(train0)+(dim(train1)[1]-1)*var(train1))/(n-2)
Sigma
```

```
## [1] "pi0 and pi1"
## [1] 0.4461538 0.5538462
## [1] "mu"
##        x1        x2
## 17.255172  5.577241
##        x1        x2
## 20.175000  2.966944
## [1] "Sigma"
##            x1         x2
## x1  7.2339559 -0.6584657
## x2 -0.6584657  0.9519388
```

- Show δ:

$$\Pr(Y=0|X=\mathbf{x}) = \Pr(Y=1|X=\mathbf{x}) = \frac{\pi_0 f_0(x)}{\pi_0 f_0(x)+\pi_1 f_1(x)} = \frac{\pi_1 f_0(1)}{\pi_0 f_0(x)+\pi_1 f_1(x)}$$

$$\pi_0 f_0(\mathbf{x}) = \pi_1 f_1(\mathbf{x}) = \pi_0 f_0(\mathbf{x}) = \pi_0 \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_0})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_0})} = \pi_1 \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_1})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_1})}$$

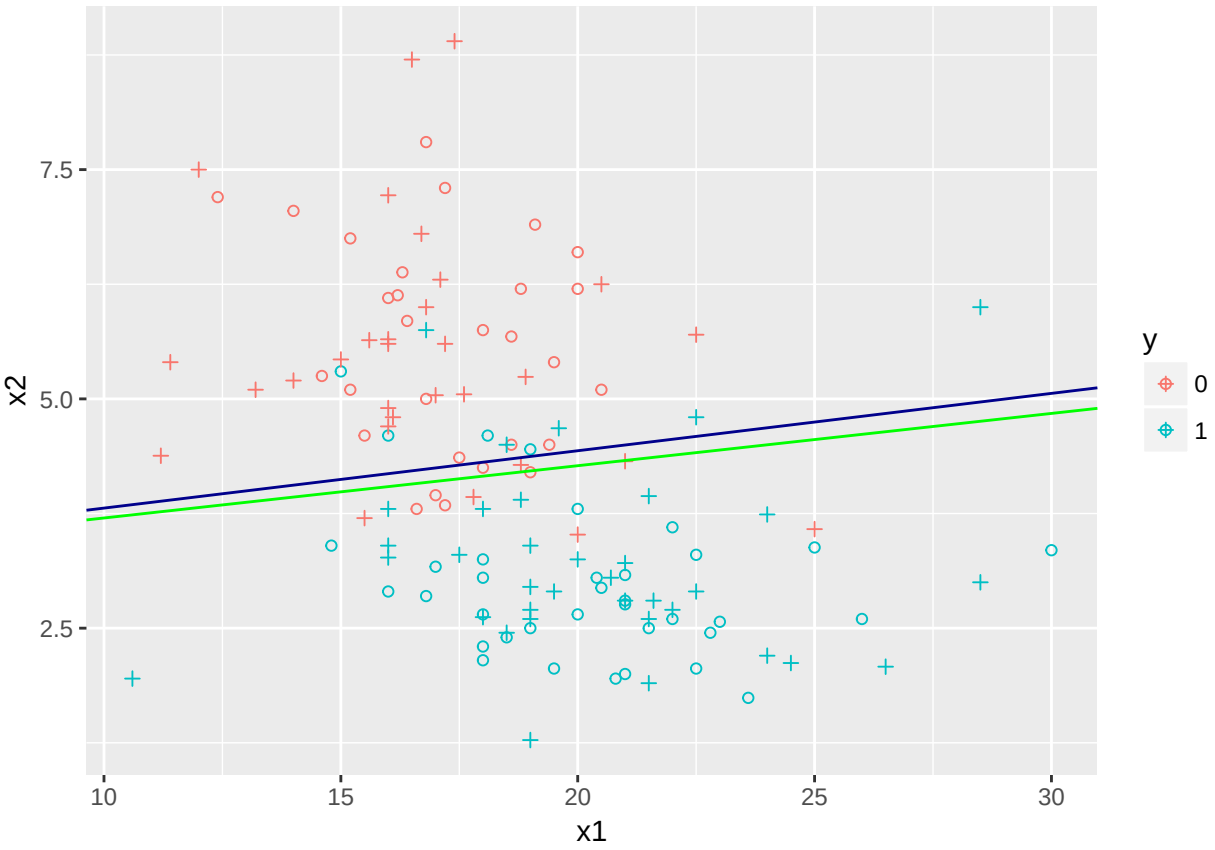
$$\log(\pi_0) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0) = \log(\pi_1) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)$$

$$\log(\pi_0) - \frac{1}{2}x^T\mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_0 - \frac{1}{2}\mu_0^T\mathbf{\Sigma}^{-1}\mu_0 = \log(\pi_1) - \frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_1 - \frac{1}{2}\mu_1^T\mathbf{\Sigma}^{-1}\mu_1$$

$$\log(\pi_0) + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_0 - \frac{1}{2}\mu_0^T\mathbf{\Sigma}^{-1}\mu_0 = \log(\pi_1) + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_1 - \frac{1}{2}\mu_1^T\mathbf{\Sigma}^{-1}\mu_1 = \delta_0(\mathbf{x}) = \delta_1($$

- LDA in R, and draw class boundary. Have added also logistic regression boundary.

```
ltrain=lda(y~x1+x2,data=train)
a=solve(Sigma)%*%(mu0-mu1)
c=-0.5*t(mu0)%*%solve(Sigma)%*%mu0+0.5*t(mu1)%*%solve(Sigma)%*%mu1+log(pi0)-log(pi1)
interceptL = -c/a[2]
slopeL = -a[1]/a[2]

g1 = ggplot(data=train,aes(x=x1, y=x2, colour=y)) + geom_point(pch = 1)
g1 + geom_point(data = test, pch = 3) + geom_abline(slope=slope,intercept=intercept,colour="green")+ ge
```



- Formula for LDA class boundary (0.5 cut-off): $\delta_0(\mathbf{x}) = \delta_1(\mathbf{x})$, and thus $\delta_0(\mathbf{x}) - \delta_1(\mathbf{x}) = 0$.

$$\log(\pi_0) + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_0 - \frac{1}{2}\mu_0^T\mathbf{\Sigma}^{-1}\mu_0 - \log(\pi_1) - \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_1 + \frac{1}{2}\mu_1^T\mathbf{\Sigma}^{-1}\mu_1 = 0$$

$$\mathbf{x}^T\mathbf{\Sigma}^{-1}(\mu_0 - \mu_1) - \frac{1}{2}\mu_0^T\mathbf{\Sigma}^{-1}\mu_0 + \frac{1}{2}\mu_1^T\mathbf{\Sigma}^{-1}\mu_1 + \log(\pi_0) - \log(\pi_1) = 0$$

  Letting all the terms except the first be noted c, then c is found in the R printout above. The equation is then of the form $ax_1 + bx_2 + c = 0$, so $x_2 = -c/b - a/bx_1$, as calculated above. This gave

```
interceptL
slopeL
```

```
##           [,1]
## [1,] 3.183897
## [1] 0.06254103
```

- Confusion table, with misclassification rate 5+6/65=0.17, sensitivity 30/35=0.86 and specificity 24/30=0.8.
- If we went from LDA to QDA we would allow the classes to have differen covariance matrices. This would give nonlinear (quadratic) class boundaries. It is not clear if that would be a good choice.

```
ltrain=lda(y~x1+x2,data=train)
lpred=predict(object = ltrain, newdata = test)$posterior[,2]
lroc=roc(response=test$y,lpred)
testclass=ifelse(lpred > 0.5, 1, 0)
t = table(test$y, testclass)
t
```

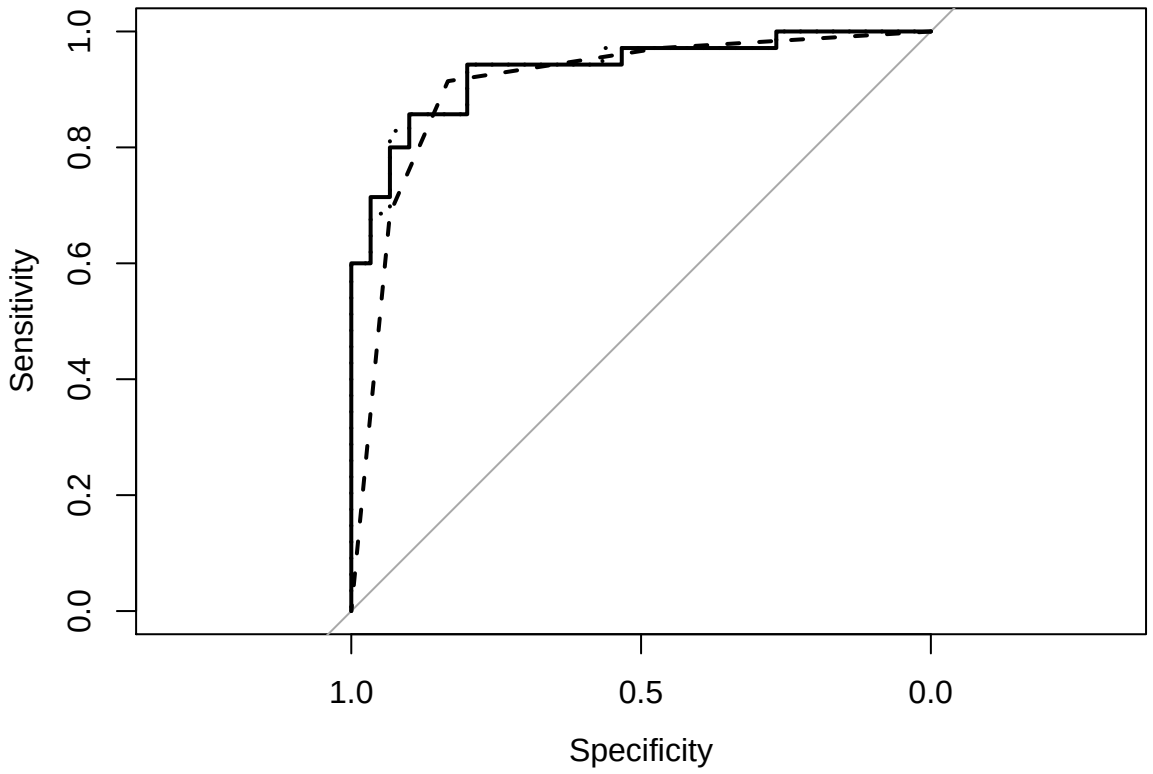
```
n = length(test$y)
error = (n-sum(diag(t)))/n
error
```

```
##    testclass
##      0  1
##   0 24  6
##   1  5 30
## [1] 0.1692308
```

d) Compare classifiers

- Compare your results from the different classification methods (logistic regression, your preferred KNN, LDA) based on the 0.5 cut-off on posterior probability classification rule. Which method would you prefer?
- Explain what an ROC curve is and why that is useful. Would your preference (to which method is the best for our data) change if a different cut-off was chosen? Answer this by producing ROC-curves for the three methods on the test set. Also calculate AUC. Hint: use function `res=roc(response=test$y,predictor)` in `library(pROC)` where the predictor is a vector with your predicted posterior probabilites for the test set, and then `plot(res)` and `auc(res)`.

```
glmroc=roc(response=test$y,predictor=predglm)
plot(glmroc)# logistic solid line
print("logistic regression")
auc(glmroc)
KNN3roc=roc(response=test$y,predictor=KNN3prob) #see above for code
plot(KNN3roc,add=TRUE,lty=2) # KNN3 dashed
print("KNN3")
auc(KNN3roc)
ltrain=lda(y~x1+x2,data=train)
lpred=predict(object = ltrain, newdata = test)$posterior[,1]
lroc=roc(response=test$y,lpred)
plot(lroc,add=TRUE,lty=3) # LDA dotted
```




```
print("LDA")
auc(lroc)
```

```
## [1] "logistic regression"
## Area under the curve: 0.9333
## [1] "KNN3"
## Area under the curve: 0.9086
## [1] "LDA"
## Area under the curve: 0.9343
```

Best method wrt AUC: LDA, but not very different from logistic regression, while 3KNN is best wrt misclassification rate on test set.

In the ROC-curve we plot the sensitivity and 1- spesificity against each other for all possible thresholds of the probability for class 1. To construct the ROC-curve we would have to calculate the sensitivity and spesificity for different values of the cutoff $p(x) > cut$. Using a threshold of 0.5, you say that if a new person has a probability of 0.51 of having the disease, he is classified as diseased. Another person with a probability of 0.49 would then be classified as non-diseased. The ROC-curve and the area under the ROC-curve are useful tools as they consider all possible thresholds for the cutoff.

The AUC is the area under the ROC-curve and gives the overall performance of the test for all possible thresholds. An AUC value of 1 means a perfect fit for all possible thresholds, while a AUC of 0.5 corresponds to the classifyer that performs no better than chance. Hence, a classification method $p(x)$ giving 0.6 and another $q(x)$ giving 0.7, we would prefer $q(x)$ as it has the highest AUC value - in general - unless there is a specific reason for only wanting to consider one specific value for the cut.off.

R-code for Problem 1

```
library(FNN)
library(ggplot2)
library(ggpubr)
library(reshape2)
maxK=25
M=100 # repeated samplings, x fixed  - examples were run with M=1000
x = seq(-3, 3, 0.1)
dfx=data.frame(x=x)
truefunc=function(x) return(-x+x^2+x^3)
true_y = truefunc(x)

set.seed(2) # to reproduce
error = matrix(rnorm(length(x)*M, mean=0, sd=2),nrow=M,byrow=TRUE)
testerror = matrix(rnorm(length(x)*M, mean=0, sd=2),nrow=M,byrow=TRUE)
ymat = matrix(rep(true_y,M),byrow=T,nrow=M) + error
testymat = matrix(rep(true_y,M),byrow=T,nrow=M) + testerror

ggplot(data=data.frame(x=x,y=ymat[1,]),aes(x,y))+geom_point(col="purple",size=2)+stat_function(fun=truefunc,lwd=1.1,colour="blac

predarray=array(NA,dim=c(M,length(x),maxK))
for (i in 1:M)
{
  for (j in 1:maxK)
  {
    predarray[i,,j]=knn.reg(train=dfx,test=dfx,y=c(ymat[i,]),k=j)$pred
  }
}
# first - just plot the fitted values - and add the true curve in black
# M curves and choose k=1,2,10,30 in KNN

# rearranging to get data frame that is useful
thislwd=1.3
stackmat=NULL
for (i in 1:M) stackmat=rbind(stackmat,cbind(x,rep(i,length(x)),predarray[i,,]))
colnames(stackmat)=c("x","rep",paste("K",1:maxK,sep=""))
sdf=as.data.frame(stackmat)
yrange=range(apply(sdf,2,range)[,3:(maxK+2)])
# making the four selected plots
p1=ggplot(data=sdf,aes(x=x,y=K1,group=rep,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
p1=p1+stat_function(fun=truefunc,lwd=thislwd,colour="black")+ggtitle("K1")
p2=ggplot(data=sdf,aes(x=x,y=K2,group=rep,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
p2=p2+stat_function(fun=truefunc,lwd=thislwd,colour="black")+ggtitle("K2")
p10=ggplot(data=sdf,aes(x=x,y=K10,group=rep,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
p10=p10+stat_function(fun=truefunc,lwd=thislwd,colour="black")+ggtitle("K10")
p25=ggplot(data=sdf,aes(x=x,y=K25,group=rep,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
p25=p25+stat_function(fun=truefunc,lwd=thislwd,colour="black")+ggtitle("K30")
ggarrange(p1,p2,p10,p25)

# calculating trainMSE and testMSE
trainMSE=matrix(ncol=maxK,nrow=M)
for (i in 1:M) trainMSE[i,]=apply((predarray[i,,]-ymat[i,])^2,2,mean)
testMSE=matrix(ncol=maxK,nrow=M)
for (i in 1:M) testMSE[i,]=apply((predarray[i,,]-testymat[i,])^2,2,mean)
#rearranging to get data frame that is useful
stackmat=NULL
for (i in 1:M) stackmat=rbind(stackmat,cbind(rep(i,maxK),1:maxK,trainMSE[i,],testMSE[i,]))
colnames(stackmat)=c("rep","K","trainMSE","testMSE")
sdf=as.data.frame(stackmat)
yrange=range(sdf[,3:4])
# plotting traning and test MSE
p1=ggplot(data=sdf[1:maxK,],aes(x=K,y=trainMSE))+scale_y_continuous(limits=yrange)+geom_line()
pall= ggplot(data=sdf,aes(x=K,group=rep,y=trainMSE,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
testp1=ggplot(data=sdf[1:maxK,],aes(x=K,y=testMSE))+scale_y_continuous(limits=yrange)+geom_line()
testpall= ggplot(data=sdf,aes(x=K,group=rep,y=testMSE,colour=rep))+scale_y_continuous(limits=yrange)+geom_line()
ggarrange(p1,pall,testp1,testpall)
```

```r
# calculating bias^2 and variance
meanmat=matrix(ncol=length(x),nrow=maxK)
varmat=matrix(ncol=length(x),nrow=maxK)
for (j in 1:maxK)
{
  meanmat[j,]=apply(predarray[,,j],2,mean) # we now take the mean over the M simulations - to mimic E and Var at each x value and
  varmat[j,]=apply(predarray[,,j],2,var)
}
bias2mat=(meanmat-matrix(rep(true_y,maxK),byrow=TRUE,nrow=maxK))^2 #here the truth is finally used!

# preparing to plot
df=data.frame(rep(x,each=maxK),rep(1:maxK,length(x)),c(bias2mat),c(varmat),rep(4,prod(dim(varmat)))) #irr is just 4
colnames(df)=c("x","K","bias2","variance","irreducible error") #suitable for plotting
df$total=df$bias2+df$variance+df$`irreducible error`
hdf=melt(df,id=c("x","K"))
# averaged over all x - to compare to train and test MSE
hdfmean=
  hdf %>%
  group_by(K,variable) %>%
  summarise (mean_value=mean(value))
ggplot(data=hdfmean[hdfmean[,1]<31,],aes(x=K,y=mean_value,colour=variable))+geom_line()+ggtitle("averaged over all x")

# extra: what about different values of x?
hdfatxa=hdf[hdf$x==-2,]
hdfatxb=hdf[hdf$x==0,]
hdfatxc=hdf[hdf$x==1,]
hdfatxd=hdf[hdf$x==2.5,]
pa=ggplot(data=hdfatxa,aes(x=K,y=value,colour=variable))+geom_line()+ggtitle("x0=-2")
pb=ggplot(data=hdfatxb,aes(x=K,y=value,colour=variable))+geom_line()+ggtitle("x0=0")
pc=ggplot(data=hdfatxc,aes(x=K,y=value,colour=variable))+geom_line()+ggtitle("x0=1")
pd=ggplot(data=hdfatxd,aes(x=K,y=value,colour=variable))+geom_line()+ggtitle("x0=2.5")
ggarrange(pa,pb,pc,pd)
```