Compulsory exercise 1, (final+added hints) version 09.02.2018

TMA4268 Statistical Learning V2018

Contact person: Martina Hall, Martina.Hall@ntnu.no

To be handed in on Blackboard: deadline February 16 at 16.00

Supervision:

- All Fridays 12-14 in Smia, in addition
- Monday February 5, 12-14 in F2,
- Wednesday February 7, 10-12 in Smia.

Maximal group size is 3 - join a group (self enroll) before handing in on Blackboard. Exercise should be handed in as one R Markdown file and a pdf-compiled version of the R Markdown file. Example file is here: https://www.math.ntnu.no/emner/TMA4268/2018v/CompEx1mal.Rmd

Maximal score is 10 points.

Problem 1 - Core concepts in statistical learning [2 points]

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 [1 point]

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?

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).

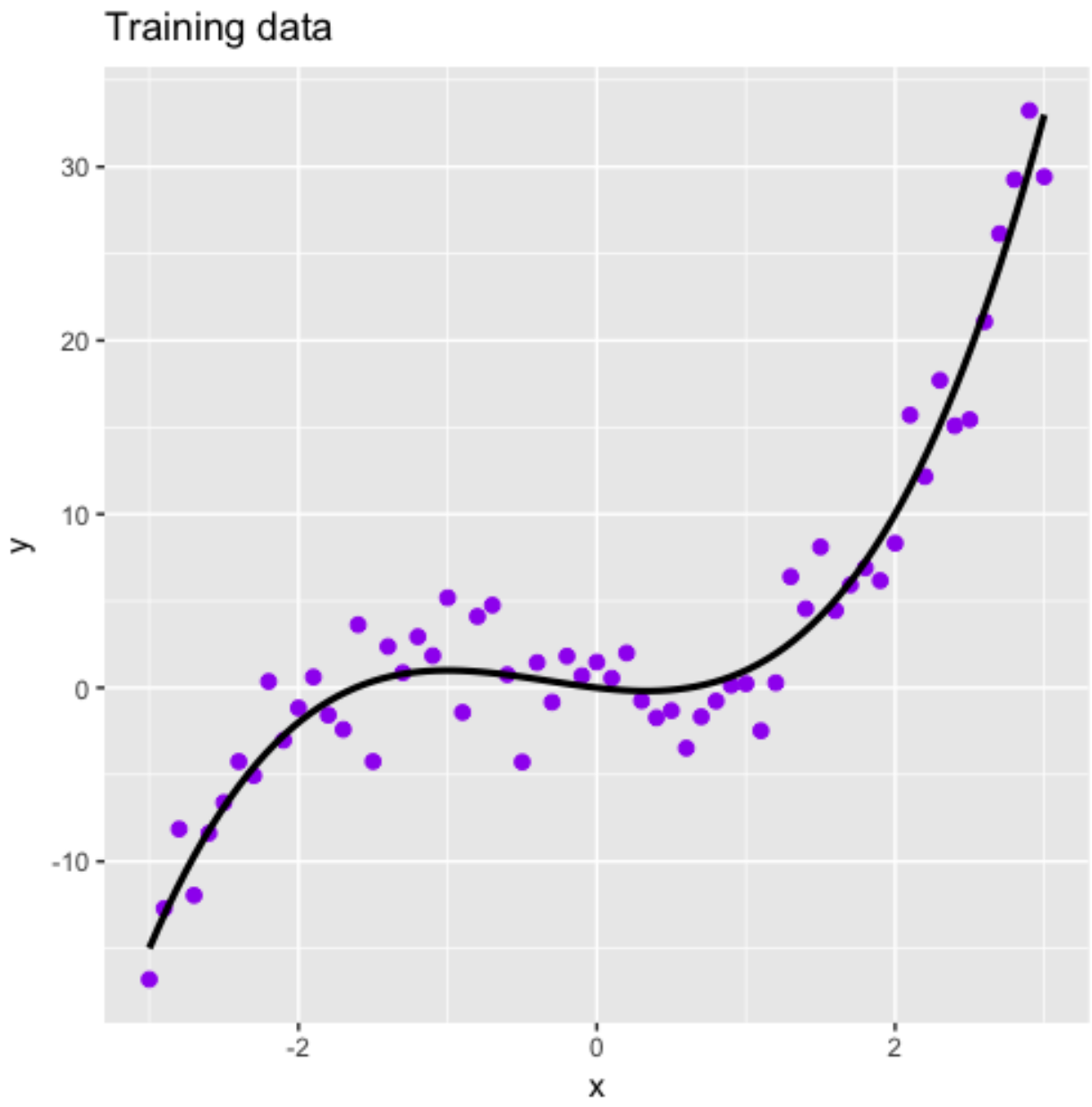


Figure 1: Figure 1

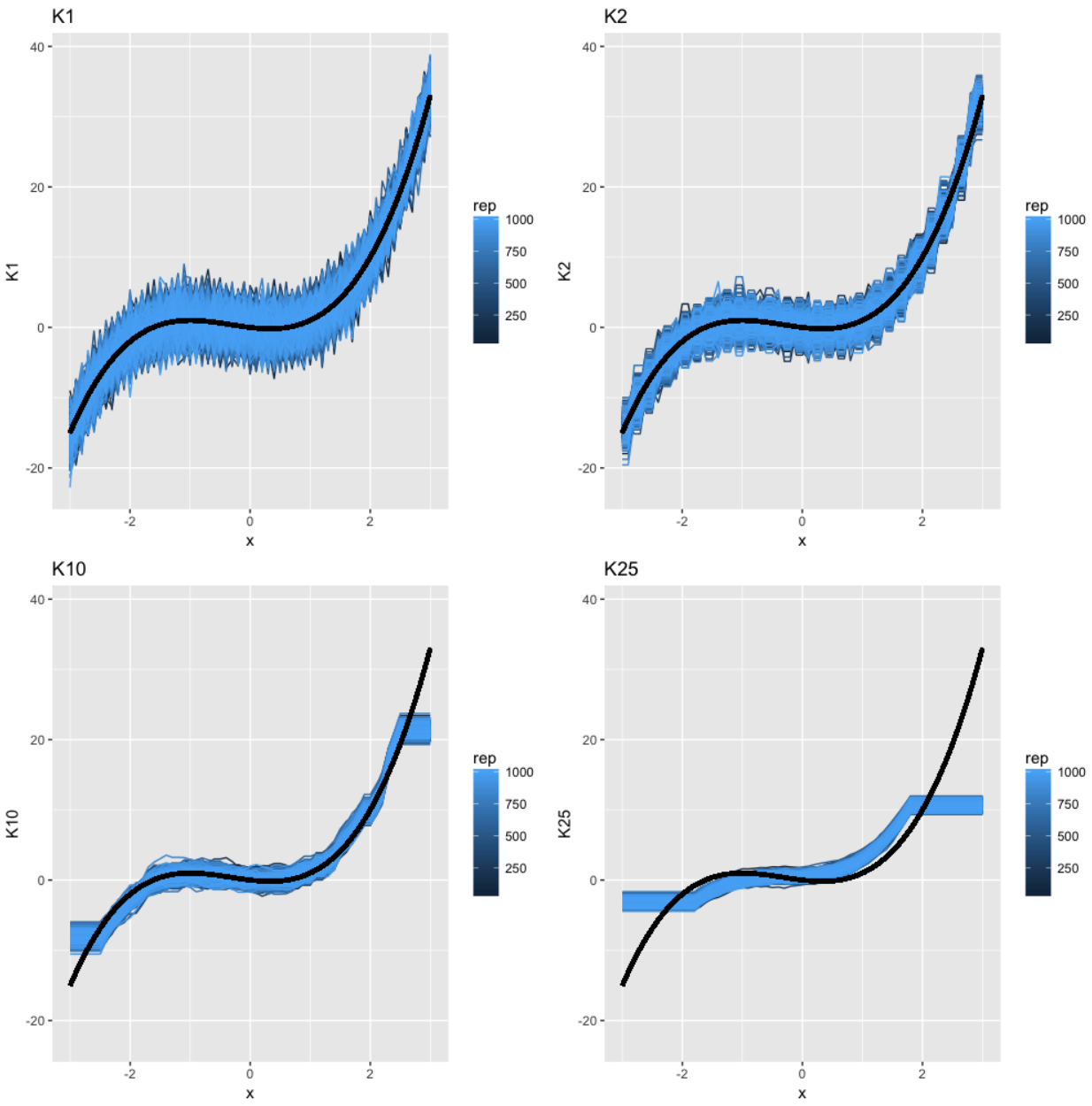


Figure 2: Figure 2

- Comment on what you see.
- What do you think is 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 [1 point]

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)?

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 [4 points]

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.

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.

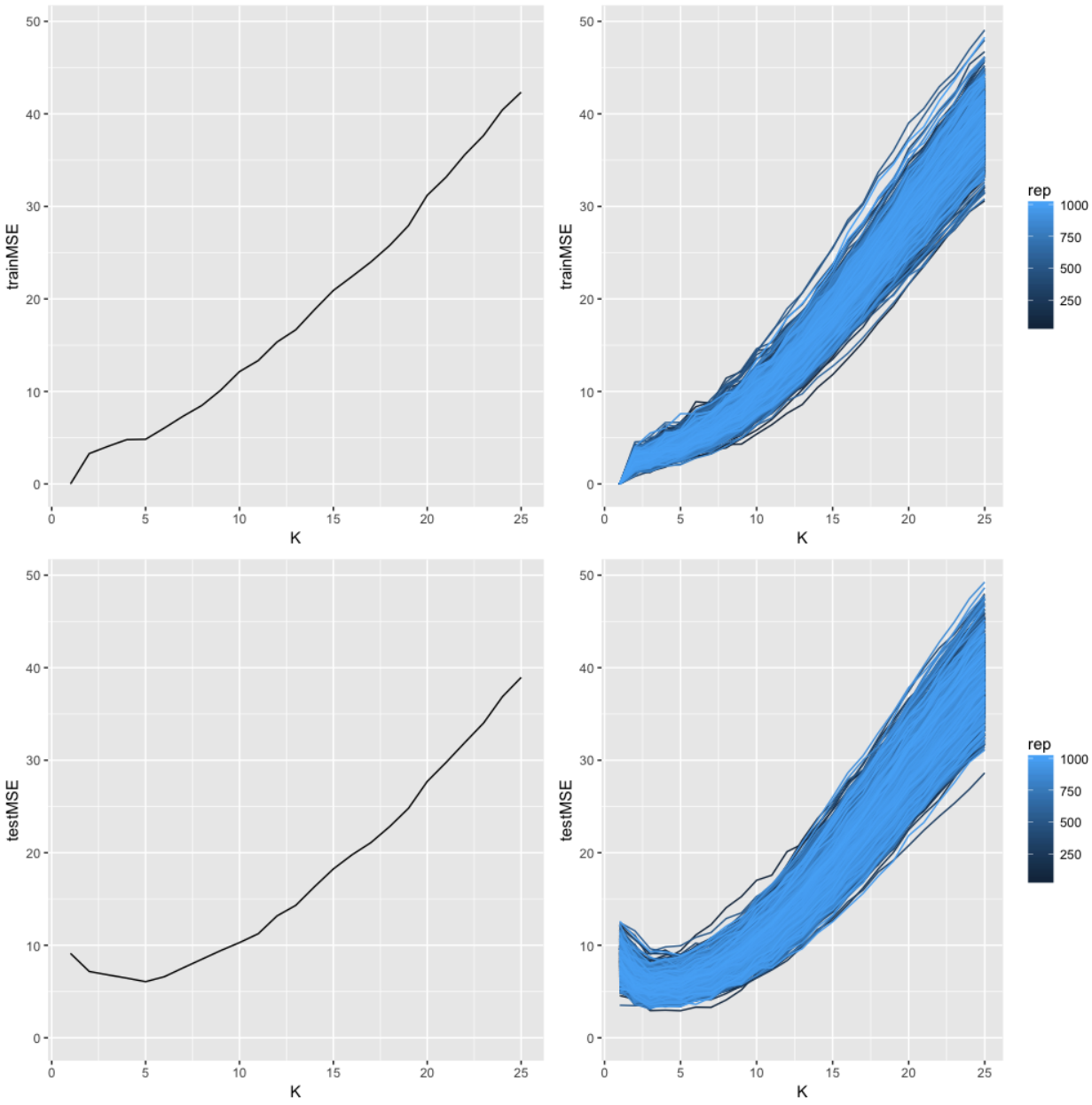


Figure 3: Figure 3

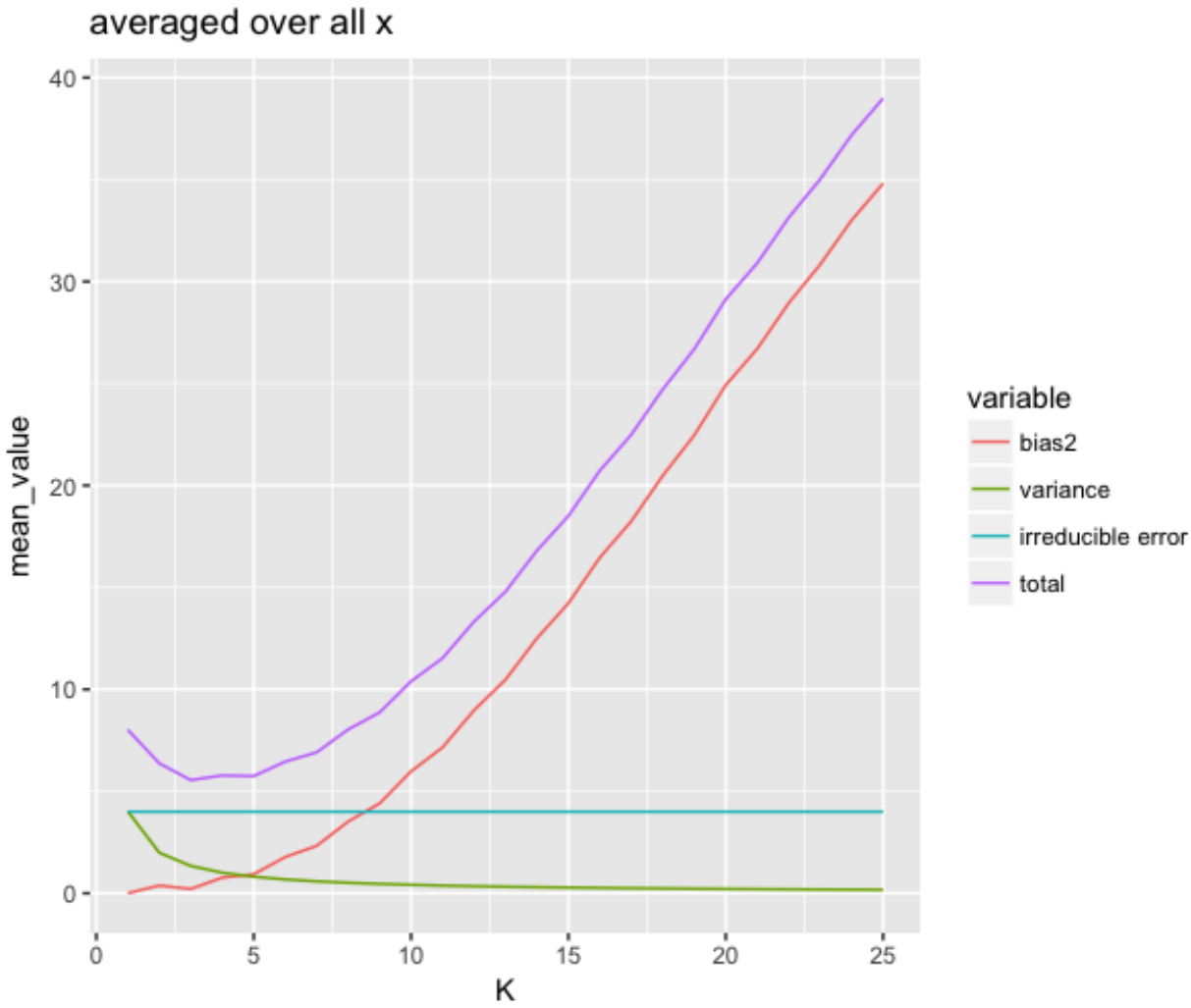


Figure 4: Figure 4

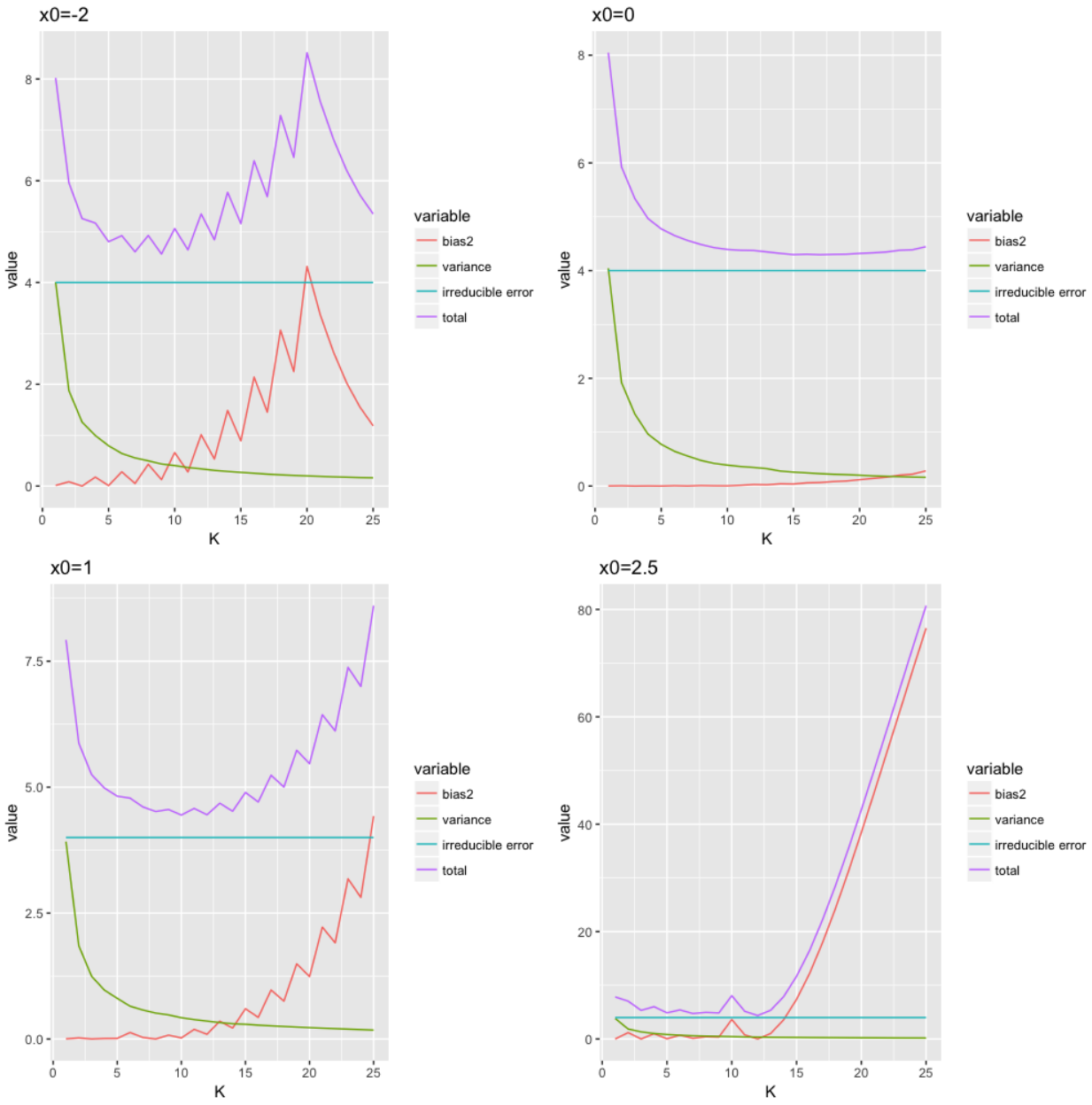


Figure 5: Figure 5

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

a) Understanding model output [1 point]

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`

b) Model fit [1 point]

- 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?

```r
# 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(

# 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 = depa

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

c) Confidence interval and hypothesis test [1 points]

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.

d) Prediction [1 point]

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)
```

- 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`.

(Comment: Is that allowed - to only do the inverse? Yes, that could be the result of a first order Taylor expansion approximation. If you think this is interesting you can read more from page 36 here: https://www.math.ntnu.no/emner/TMA4267/2017v/L12.pdf - but that is not on the reading list of this course. You have probably used this result in your physics courses.)

- 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.

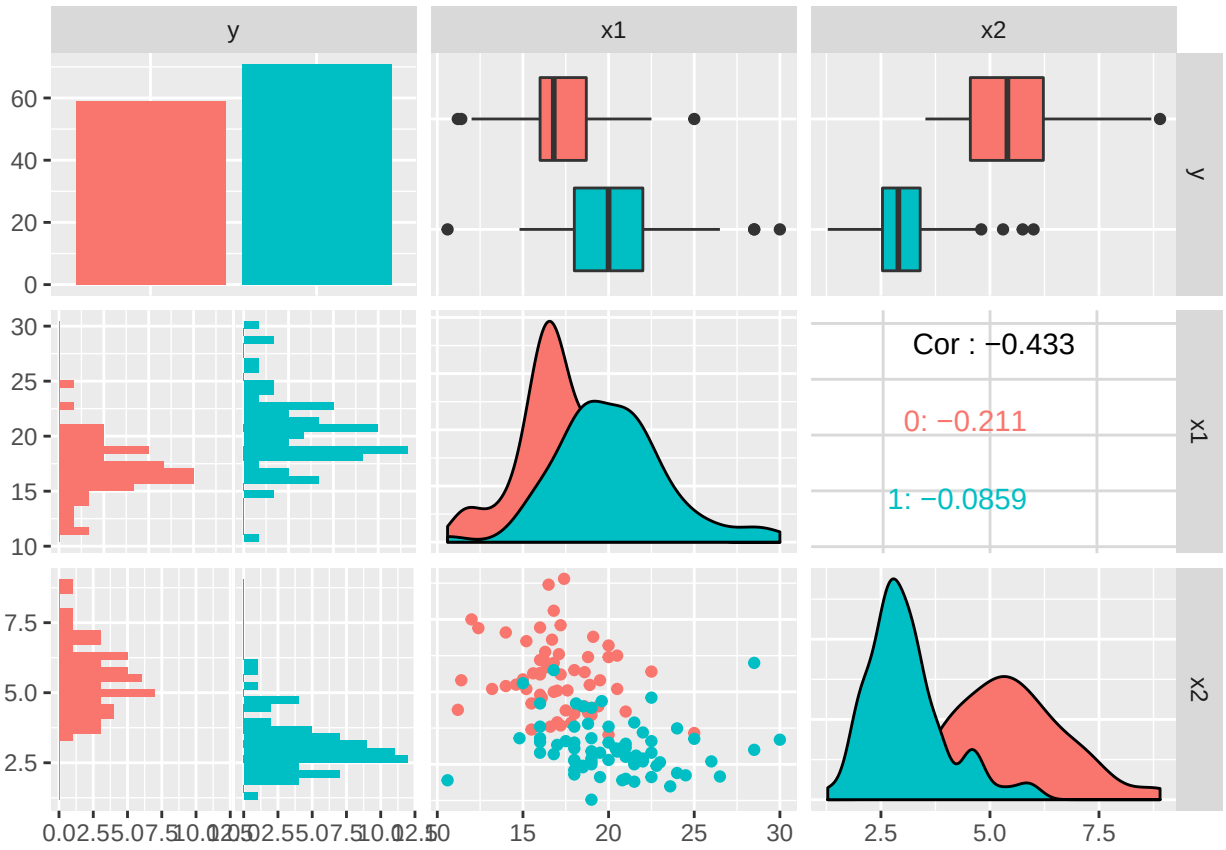
Problem 3 - Classification [4 points]

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$class=as.factor(wine$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 [1 point]

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 $\hat{\beta}_2$. (Hint: odds.)
- 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? (Hint: write out the equation for the boundary as $x_2 = intercept + slope * x_1$)
- 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`.

Hint: if you have your calulated the slope and the intercept, your plot of train and test with boundary can be produced with:

```
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)+ggtitle("train and
```

- 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?

Hint: if you have the predicted probabilities for class 1 for the test set in `pred` then you may do this to produce the confusion matrix:

```
testclass=ifelse(pred > 0.5, 1, 0)
t = table(test$y, testclass)
t
```

b) K-nearest neighbor classifier [0.5 point]

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

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

- 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?

Hint: you may not need this before d) if you just use the classification (and not the probabilites) when using KNN on the test set, but there is a slight complication on the output from prediction with KNN, since the probability of the *winning class* is given. Here is some code to avoid complications.

```
KNN3 = knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = F)
KNN3probwinning = attributes(knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = TRUE
KNN3prob <- ifelse(KNN3 == "0", 1-KNN3probwinning, KNN3probwinning)
# now KNN3prob is the probability of class 1 - and can be used for roc in d)
```

c) LDA (& QDA) [1.5 point]

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

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

- We use the LDA 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.
- Perform LDA on the training data (using R). Just show the R-code for doing this
- Make the confusion table for the test set when using 0.5 as cut-off (Hint: just use predict on your lda-object). 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?

d) Compare classifiers [1 point]

- 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)`.

Hint: here is the code - you focus on interpretation!

```
library(pROC)
glmroc=roc(response=test$y,predictor=predglm)
plot(glmroc)
auc(glmroc)
KNN3 = knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = F)
KNN3probwinning = attributes(knn(train = train[,-1], test = test[,-1], k = 3, cl = train$y, prob = TRUE
KNN3prob <- ifelse(KNN3 == "0", 1-KNN3probwinning, KNN3probwinning)
KNN3roc=roc(response=test$y,predictor=KNN3prob)
plot(KNN3roc)
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)
auc(lroc)
```

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)
```