

Øving 1

2022-04-21

Problem 2

ref: Figure 2.9 (p.31)

- a) Discuss whether a flexible or rigid method typically will have the highest test error.

A: We see that the test error changes as an U-shape relative to flexibility. This means that both a very inflexible and a highly flexible model will have a high test error. This is explained by underfitting (too simple model) and overfitting (too complex model, adjusting to much to the noise in the data). Therefore we want to choose something in the middle, usually around the minimum of the test MSE. The training MSE will always decrease when we have a more flexible model.

- b) Does a small variance imply that the data has been under or overfit?

Underfit. Small variance in a model means that the model doesn't change much when we change the training and test samples.

- c) Relate the problem of over- and underfitting to the bias-variance trade-off.

Usually a underfitted model has a higher bias and a lower variance. Lower variance is explained by the model not changing much depending on the choice of the test and training split. The bias is usually quite high because the model makes assumptions about the data that might not be true (using a linear model for nonlinear data). Underfitting can also happen due to a low amount of data.

Overfitted models usually has a high variance and a low bias. The high variance means that the model varies a lot depending on what data is used for training. This overfitting can be explained by the model being too adjusted to the noise in the data, ignoring the important patterns and being too concerned with noise. The low bias comes from the model not making as many assumptions and being more flexible. This is usually good for nonlinear data.

The optimal model is a model that has low variance and low bias - that's why the bias-variance trade-off is so important.

Problem 3

```
library(ISLR)
data(Auto)
```

- a) View the data, what are the dimensions of the data? Which predictors are quantitative and qualitative?

```
dim(Auto)
```

```
## [1] 392 9
```

We have 392 samples, with 9 variables. 8 predictors, 1 response. 392 rows, 9 columns.

```
summary(Auto)
```

```
##      mpg      cylinders displacement  horsepower      weight
## Min.   : 9.00   Min.    :3.000   Min.    : 68.0   Min.    : 46.0   Min.    :1613
## 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
## Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
## Mean   :23.45   Mean    :5.472   Mean    :194.4   Mean    :104.5   Mean    :2978
## 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
## Max.   :46.60   Max.    :8.000   Max.    :455.0   Max.    :230.0   Max.    :5140
##
## acceleration      year      origin      name
## Min.    : 8.00   Min.    :70.00   Min.    :1.000   amc matador      : 5
## 1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto       : 5
## Median :15.50   Median :76.00   Median :1.000   toyota corolla   : 5
## Mean    :15.54   Mean     :75.98   Mean     :1.577   amc gremlin      : 4
## 3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet       : 4
## Max.    :24.80   Max.     :82.00   Max.     :3.000   chevrolet chevette: 4
##                                     (Other)           :365
```

mpg, cylinders, displacement, horsepower, weight, acceleration, year are quantitative variables (1-7). Origin (number between 1 and 3) and name are qualitative variables.

b) What is the range (min, max) of each quantitative predictor?

```
sapply(Auto[,seq(1:7)], range)
```

```
##      mpg cylinders displacement horsepower weight acceleration year
## [1,]  9.0         3          68         46    1613          8.0    70
## [2,] 46.6         8         455        230    5140         24.8    82
```

The output answers the question. Minimum value is the first, maximum value the last.

c) What is the mean and standard deviation of each quantitative predictor?

```
#The mean values of each quantitative predictor
sapply(Auto[,seq(1:7)], mean)
```

```
##      mpg cylinders displacement horsepower      weight acceleration
## 23.445918  5.471939  194.411990  104.469388 2977.584184  15.541327
##      year
## 75.979592
```

```
#The standard deviations of each quantitative predictor
sapply(Auto[,seq(1:7)], sd)
```

```
##      mpg cylinders displacement horsepower      weight acceleration
## 7.805007  1.705783  104.644004  38.491160  849.402560  2.758864
##      year
## 3.683737
```

- d) Now, make a new dataset called ReducedAuto where you removed the 10th through 85th observations. What is the range, mean and standard deviation of the quantitative predictors in this reduced set?

```
ReducedAuto = Auto[-seq(10:85),]  
dim(ReducedAuto)
```

```
## [1] 316 9
```

As the dimensions of the rows is reduced by 84, we did the correct reduction.

```
#The ranges of each quantitative predictor in the reduced dataset  
sapply(ReducedAuto[,seq(1:7)], range)
```

```
##      mpg cylinders displacement horsepower weight acceleration year  
## [1,] 11.0         3          68         46    1649          9.5    72  
## [2,] 46.6         8         455        230    4997         24.8    82
```

```
#The mean values of each quantitative predictor in the reduced dataset  
sapply(ReducedAuto[,seq(1:7)], mean)
```

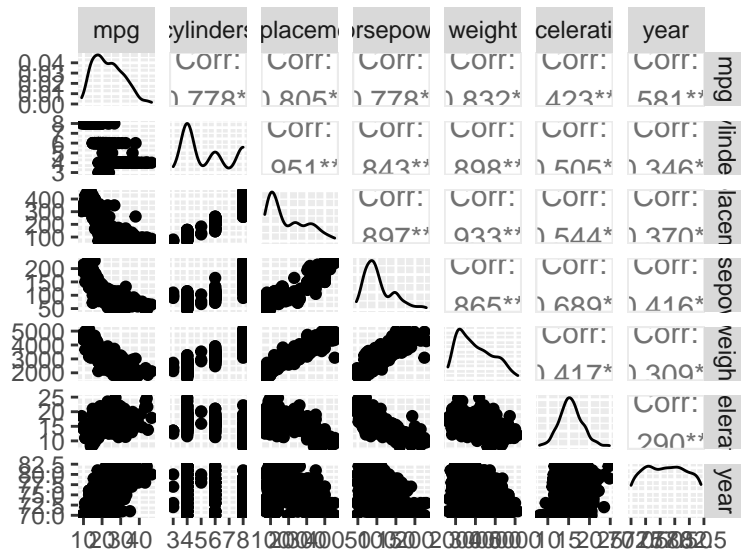
```
##      mpg      cylinders displacement      horsepower      weight acceleration  
## 24.622785  5.272152  180.474684    98.370253 2898.898734    15.894620  
##      year  
## 77.205696
```

```
#The standard deviation of each quantitative predictor in the reduced dataset  
sapply(ReducedAuto[,seq(1:7)], sd)
```

```
##      mpg      cylinders displacement      horsepower      weight acceleration  
##  7.758820  1.612053   94.987598    33.072968   799.676920    2.554014  
##      year  
##  2.985483
```

- e) Using the full dataset, investigate the quantitative predictors graphically using a scatterplot. Do you see any strong relationships between the predictors?

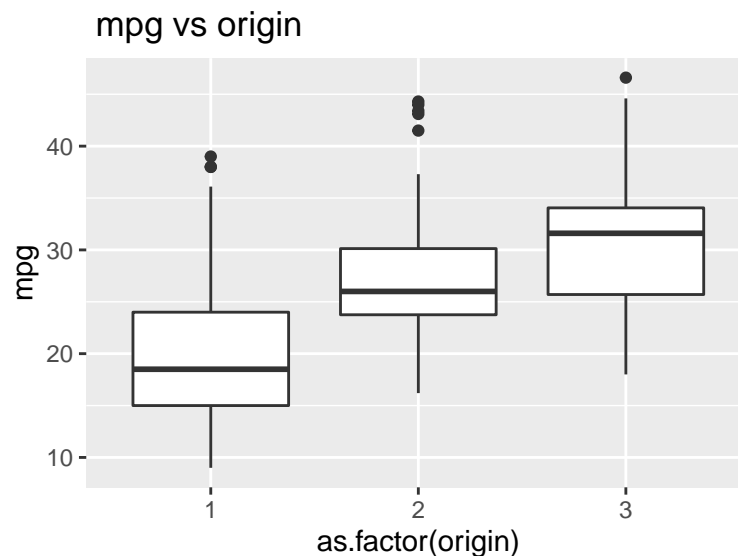
```
library(GGally)  
quantitative_data = Auto[,seq(1:7)]  
ggpairs(quantitative_data)
```



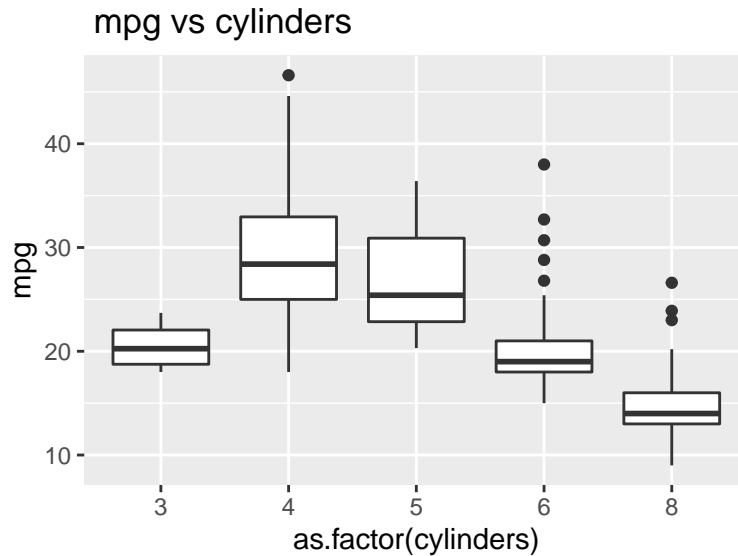
Comment: There seems to be a strong relationship between mpg and weight, mpg and displacement, mpg and horsepower.

- f) Suppose we wish to predict gas mileage (mpg) on the basis of other variables. Make some plots showing the relationship between the mpg and the qualitative variables. Which predictors would you consider helpful when predicting mpg?

```
ggplot(Auto, aes(as.factor(origin), mpg)) + geom_boxplot() + labs(title = " mpg vs origin")
```



```
ggplot(Auto, aes(as.factor(cylinders), mpg)) + geom_boxplot() + labs(title = " mpg vs cylinders")
```



From the first plot, there seems to be a strong relationship between mpg and weight, displacement and horsepower.

The boxplots tells us that the 3rd origin has the highest mpg, and the 1st origin has the lowest. There is a clear dependence on this variable. The same goes for cylinders.

In conclusion, I would use weight, displacement, horsepower, cylinders and origin as predictors for mpg.

- g) Use only the covariance matrix to find the correlation between mpg and displacement, mpg and horsepower, and mpg and weight.

Does it coincide with correlation matrix using `cor()`?

#Using the built-in method

```
cor(Auto[,seq(1:7)])
```

```
##           mpg  cylinders displacement horsepower   weight
## mpg          1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year         0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
##
##      acceleration      year
## mpg          0.4233285  0.5805410
## cylinders    -0.5046834 -0.3456474
## displacement -0.5438005 -0.3698552
## horsepower   -0.6891955 -0.4163615
## weight       -0.4168392 -0.3091199
## acceleration  1.0000000  0.2903161
## year         0.2903161  1.0000000
```

#Using the built-in method

```
cov_mat = cov(Auto[,seq(1:7)])
cov_mat
```

```
##           mpg  cylinders displacement  horsepower  weight
## mpg      60.918142 -10.352928   -657.5852  -233.85793 -5517.4407
## cylinders -10.352928   2.909696    169.7219   55.34824  1300.4244
## displacement -657.585207 169.721949 10950.3676 3614.03374 82929.1001
## horsepower -233.857926  55.348244   3614.0337 1481.56939 28265.6202
## weight     -5517.440704 1300.424363 82929.1001 28265.62023 721484.7090
## acceleration  9.115514  -2.375052   -156.9944  -73.18697  -976.8153
## year        16.691477  -2.171930   -142.5721  -59.03643  -967.2285
##           acceleration  year
## mpg      9.115514  16.691477
## cylinders -2.375052  -2.171930
## displacement -156.994435 -142.572133
## horsepower  -73.186967  -59.036432
## weight     -976.815253 -967.228457
## acceleration  7.611331   2.950462
## year        2.950462  13.569915
```

#Using the built-in method

```
for(i in 1:7){
  for(j in 1:7){
    cov_mat[i,j] = cov_mat[i,j]/(sqrt(cov_mat[i,i])*sqrt(cov_mat[j,j]))
  }
}

print(cov_mat)
```

```
##           mpg  cylinders displacement horsepower  weight
## mpg      1.000000 -6.0693105   -6.284022  -6.075627  -6.495672
## cylinders -6.069310  1.0000000    1.621898   1.437947   1.530987
## displacement -6.284022  1.6218985    1.000000  93.892565  97.632270
## horsepower  -6.075627  1.4379469    93.892565   1.000000  33.277060
## weight     -6.495672  1.5309871    97.632270  33.277060   1.000000
## acceleration  3.304082 -0.8608805   -56.905461 -26.527935 -354.064285
## year        4.531127 -0.5895996   -38.703130 -16.026236 -262.567218
##           acceleration  year
## mpg      3.3040824  4.5311266
## cylinders -0.8608805  -0.5895996
## displacement -56.9054613 -38.7031297
## horsepower  -26.5279346 -16.0262362
## weight     -354.0642853 -262.5672181
## acceleration  1.0000000  0.8009427
## year        0.8009427  1.0000000
```

Here we used $cor(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$ to find the correlation matrix.

Problem 4

- a) Use the `mvnrm()` function from the MASS library to simulate 1000 values from multivariate normal distribution with (see task sheet).

```
library(MASS)

#Creating the mu and the sigmas

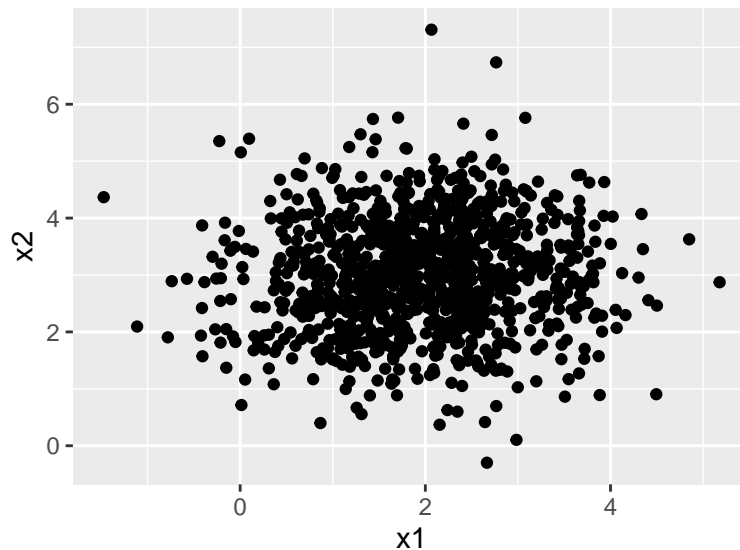
mu= c(2,3)

sigma1 = matrix(data = c(1,0,0,1), nrow = 2, ncol = 2, byrow = FALSE)
sigma2 = matrix(data = c(1,0,0,5), nrow = 2, ncol = 2, byrow = FALSE)
sigma3 = matrix(data = c(1,2,2,5), nrow = 2, ncol = 2, byrow = FALSE)
sigma4 = matrix(data = c(1,-2,-2,5), nrow = 2, ncol = 2, byrow = FALSE)

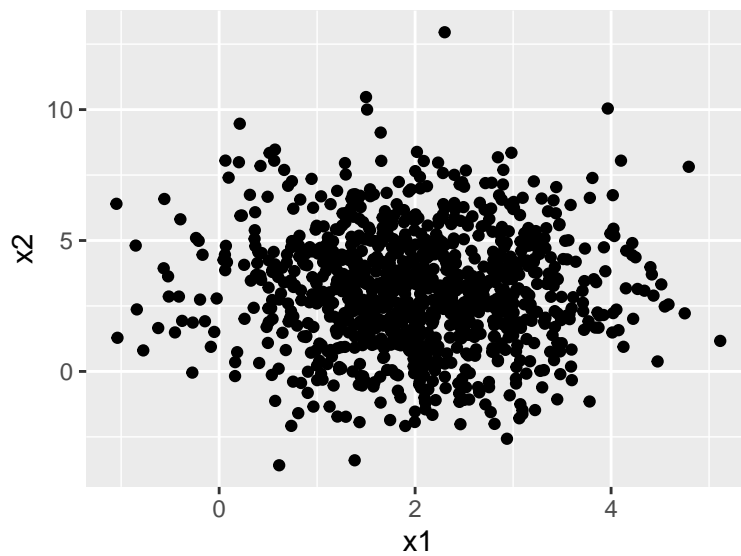
mod1 = as.data.frame(mvnrm(n = 1000, mu=mu, Sigma = sigma1))
mod2 = as.data.frame(mvnrm(n = 1000, mu=mu, Sigma = sigma2))
mod3 = as.data.frame(mvnrm(n = 1000, mu=mu, Sigma = sigma3))
mod4 = as.data.frame(mvnrm(n = 1000, mu=mu, Sigma = sigma4))
```

- b) Make a scatterplot of the four sets of simulated datasets. Can you see which plot belongs to which distribution?

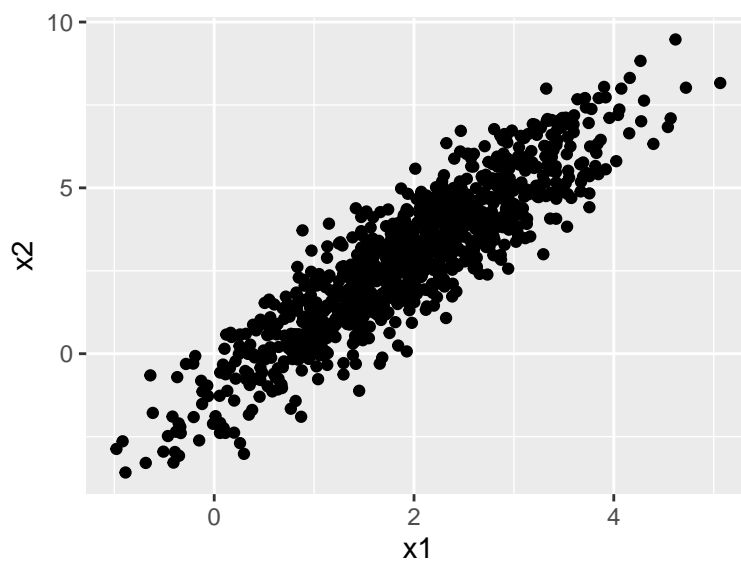
```
#First plot
colnames(mod1) = c("x1", "x2")
ggplot(mod1, aes(x1,x2)) + geom_point()
```



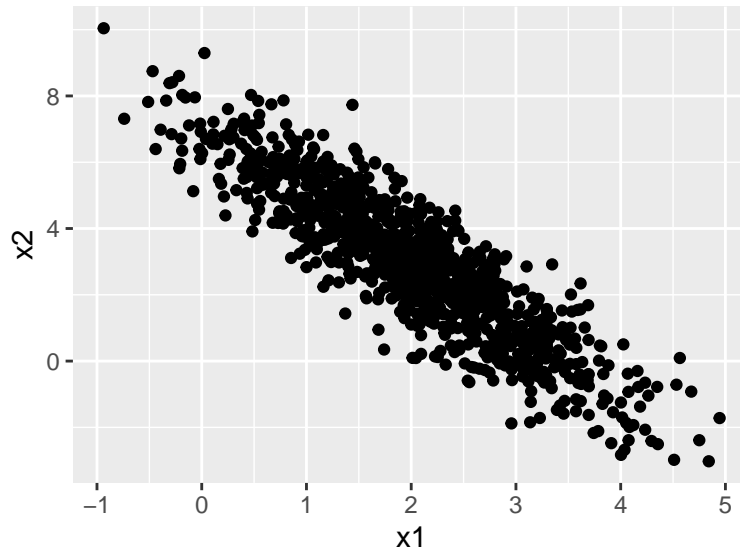
```
#Second plot
colnames(mod2) = c("x1", "x2")
ggplot(mod2, aes(x1,x2)) + geom_point()
```



```
#3rd plot
colnames(mod3) = c("x1", "x2")
ggplot(mod3, aes(x1, x2)) + geom_point()
```



```
#4th plot
colnames(mod4) = c("x1", "x2")
ggplot(mod4, aes(x1, x2)) + geom_point()
```

Problem 5

```

set.seed(2) #To reproduce

M <- 100 #Repeated samplings, x fixed

nord <- 20 #Order of polynomials

x <- seq(-2,4, 0.1) #Numbers between -2 and 4, incrementing by 0.1

#True function, x^2

true_func <- function(x){
  return (x^2)
}

true_y = true_func(x) #True y-values

error <- matrix(rnorm(length(x) * M, mean = 0 , sd = 2), nrow = M, byrow = TRUE)

y_mat <- matrix(rep(true_y, M), byrow = T, nrow = M) + error #Each row is a simulation

predictions_list <- lapply(1:nord, matrix, data = NA, nrow = M, ncol = ncol(y_mat))

for(i in 1:nord){
  for(j in 1:M){
    predictions_list[[i]][j,] <- predict(lm(y_mat[j,] ~ poly(x,i, raw = TRUE)))
  }
}

#install.packages("tidyverse")

library(tidyverse)

ist_of_matrices_with_deg_id <- lapply(1:nord, function(poly_degree) cbind(predictions_list[[poly_degree]]

```