Recommended Exercises (Module 2)

Jim Totland

1/16/2021

Link to problem set

Problem 1

a) Weather forecasting. Response: "Sunny", "Cloudy", "Rain" etc. Predictors: Air pressure, temperature, and the weather of the previous day(s). The goal is to predict.

b) Battery life of a phone. Response: Time until the phone is dead. Predictors: Screen size, Battery specs, Processor etc. Both prediction and inference are relevant here. Given a phone, we want to be able to predict what the battery life will be, based to the predictors, but from the regression we will also be able to infer which predictors are most significant.

Problem 2

a) In this example, the more flexible methods have a smaller test MSE. But at some point the test MSE start to increase monotonically with the flexibility. This is a result of overfitting.

b) The variance refers to how much $\hat{f}$ would change if we used another set of training data. A small variance could indicate that a rigid method has been used, implying that the data is most likely underfiited.

c) Bias generally decreases with flexibility, which indicates that a very low bias is connected to overfitting the data.

Problem 3

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

a) Use the `glimpse` function from the tidyverse:

```r
glimpse(Auto)
```

```
## Rows: 392
## Columns: 9
## $ mpg          <dbl> 18, 15, 18, 16, 17, 15, 14, 14, 14, 15, 15, 14, 15, 14...
## $ cylinders    <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 6, 6, 6, ...
## $ displacement <dbl> 307, 350, 318, 304, 302, 429, 454, 440, 455, 390, 383,...
## $ horsepower   <dbl> 130, 165, 150, 150, 140, 198, 220, 215, 225, 190, 170,...
## $ weight       <dbl> 3504, 3693, 3436, 3433, 3449, 4341, 4354, 4312, 4425, ...
## $ acceleration <dbl> 12.0, 11.5, 11.0, 12.0, 10.5, 10.0, 9.0, 8.5, 10.0, 8....
## $ year         <dbl> 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70...
## $ origin       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ...
## $ name         <fct> chevrolet chevelle malibu, buick skylark 320, plymouth...
```

The data has dimensions 392×9. All predictors except `name` are quantitative, although some of the them may also be treated as categorical.

b) The range is found by applying the `range()` function. For example:

```
range(Auto$mpg)
```

```
## [1]  9.0 46.6
```

Alternatively use `sapply`:

```
quant = c(1,3,4,5,6,7)
sapply(Auto[, quant], range)
```

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

c) The mean and standard deviation can be found in the following way:

```
for (i in 1:8) {
  print(summarise(Auto, mean = mean(Auto[,i]), sd = sd(Auto[,i])))
}
```

```
##       mean       sd
## 1 23.44592 7.805007
##       mean       sd
## 1 5.471939 1.705783
##      mean      sd
## 1 194.412 104.644
##       mean       sd
## 1 104.4694 38.49116
##       mean       sd
## 1 2977.584 849.4026
##       mean       sd
## 1 15.54133 2.758864
##       mean       sd
## 1 75.97959 3.683737
##       mean        sd
## 1 1.576531 0.8055182
```

d) Possible, though not very clean, solution:

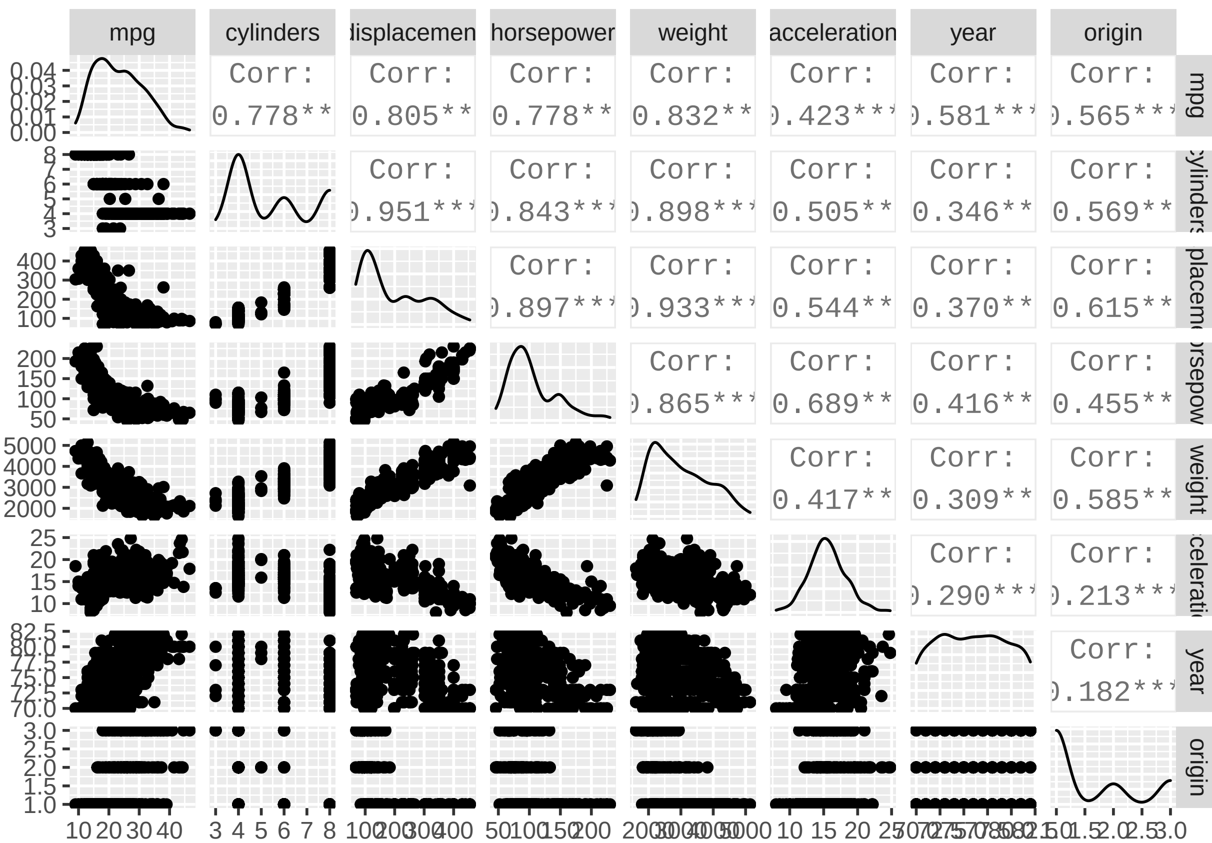
```
ReducedAuto <- Auto[- (10:85),]

for (i in 1:8) {
  print(summarise(ReducedAuto, mean = mean(ReducedAuto[,i]),
                  sd = sd(ReducedAuto[,i]),
                  range = range(ReducedAuto[,i])))
}
```

```
##       mean       sd range
## 1 24.40443 7.867283  11.0
## 2 24.40443 7.867283  46.6
##       mean       sd range
## 1 5.373418 1.654179     3
## 2 5.373418 1.654179     8
##       mean       sd range
## 1 187.2405 99.67837    68
```

```
## 2 187.2405 99.67837   455
##        mean        sd range
## 1 100.7215 35.70885    46
## 2 100.7215 35.70885   230
##        mean        sd range
## 1 2935.972 811.3002  1649
## 2 2935.972 811.3002  4997
##       mean       sd range
## 1 15.7269 2.693721   8.5
## 2 15.7269 2.693721  24.8
##        mean       sd range
## 1 77.14557 3.106217    70
## 2 77.14557 3.106217    82
##        mean      sd range
## 1 1.601266 0.81991     1
## 2 1.601266 0.81991     3
```

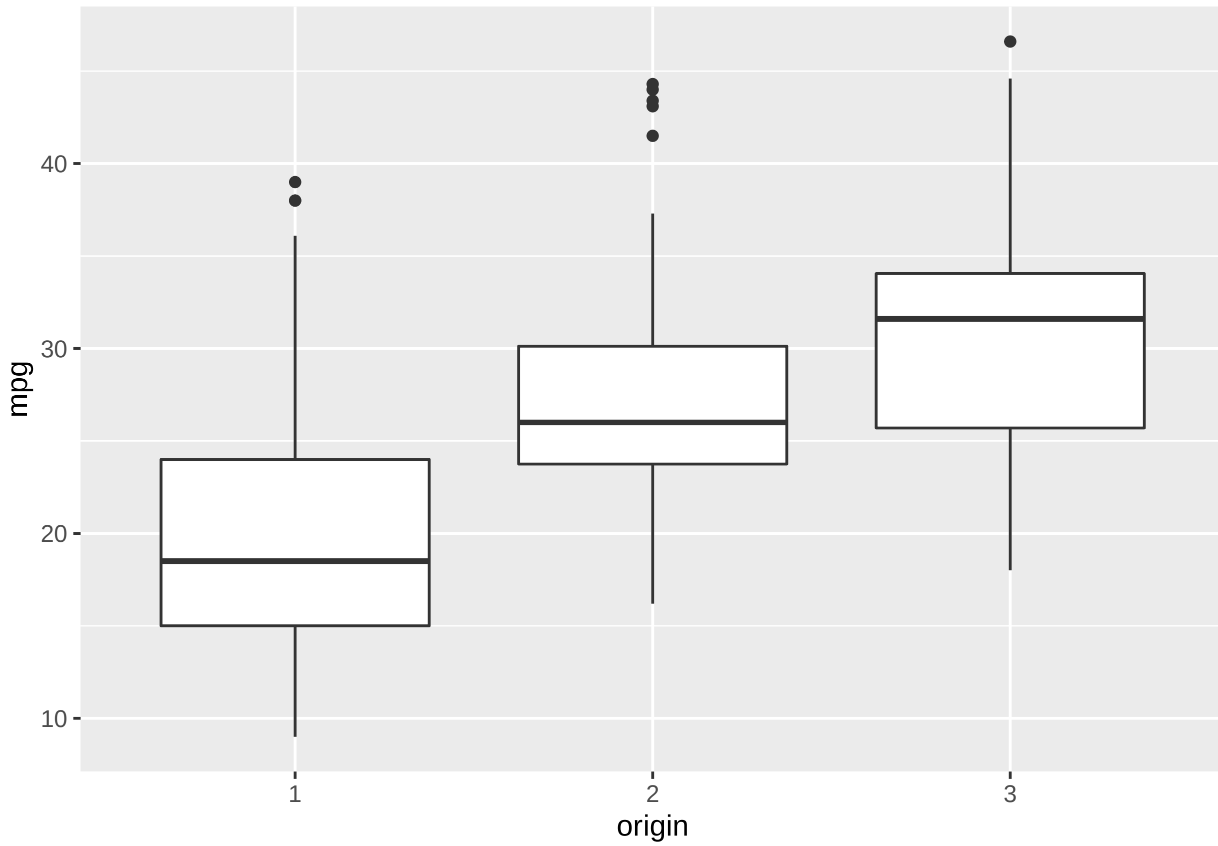
e)

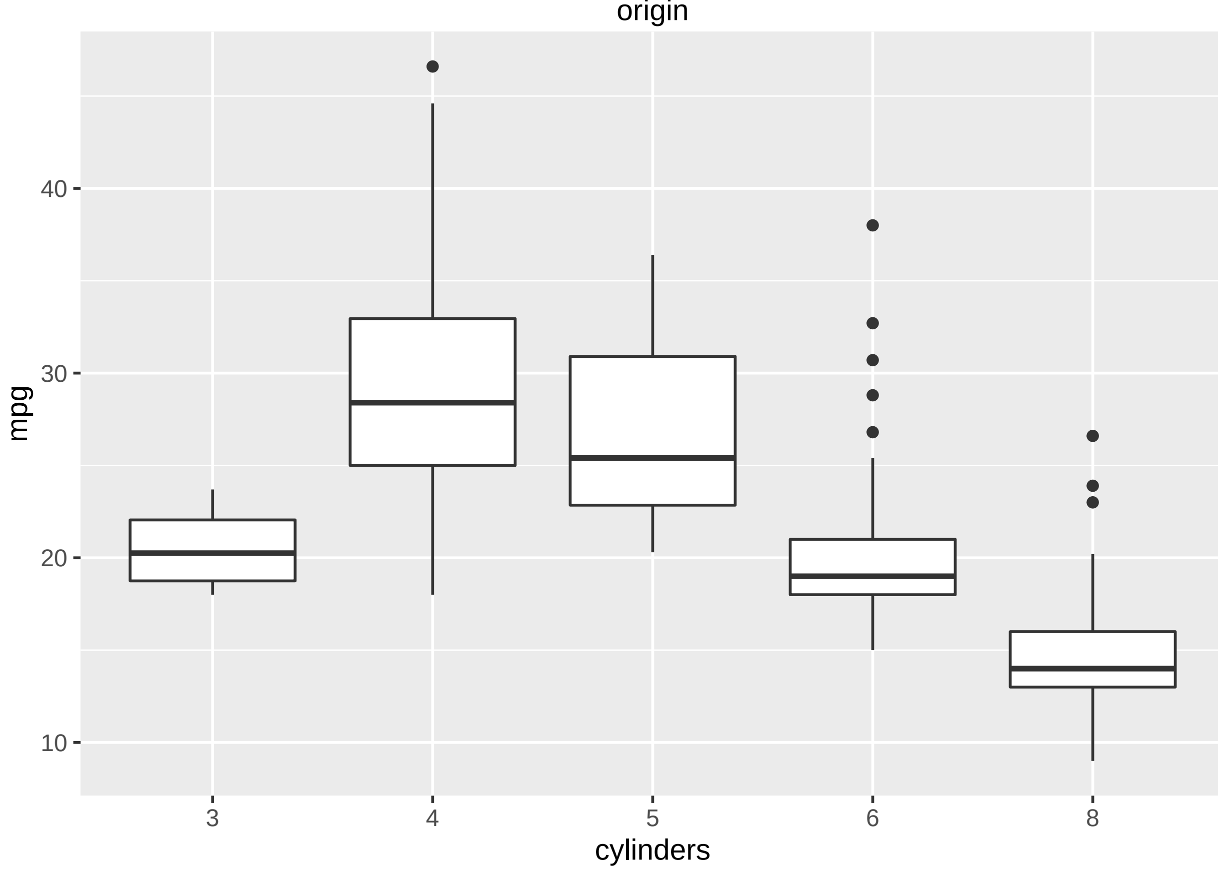
```r
library(GGally)
ggpairs(Auto[-9])
```



From the plot we can see that there seems to be a linear relationship between multiple predictors. E.g. `weight` and `displacement` have a clearly positive linear relationship. There also seems to be some non-linear relationships, e.g. between `mpg` and `horsepower`.

f) I will here treat `cylinders` and `origin` as qualitative variables and get the following box plots:





The majority of the variables seem to have some relvance in predicrting `mpg`. But the variables `year`, `acceleration` and `name` are probably the least impactful based on visual inspection.

g) The following function calculates the correlation matrix given the covariance matrix.

```
getCor <- function(covMat) {
  rows <- dim(covMat)[1]
  cols <- dim(covMat)[2]
  corMat <- matrix(nrow = rows, ncol = cols)

  for (i in 1:rows) {
    for(j in 1:cols) {
      corMat[i,j] = covMat[i,j] / (sqrt(covMat[i,i]) * sqrt(covMat[j,j]))
    }
  }
  return (corMat)
}
```

Problem 4

a)

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
v1 <- as.data.frame(mvrnorm(n = 1000,
              mu = c(2,3),
              Sigma = matrix(c(1, 0, 0, 1), 2, 2, byrow = T) ))
colnames(v1) = c("x1", "x2")

v2 <- as.data.frame(mvrnorm(n = 1000,
              mu = c(2,3),
              Sigma = matrix(c(1, 0, 0, 5), 2, 2, byrow = T) ))
colnames(v2) = c("x1", "x2")

v3 <- as.data.frame(mvrnorm(n = 1000,
              mu = c(2,3),
              Sigma = matrix(c(1, 2, 2, 5), 2, 2, byrow = T) ))
colnames(v3) = c("x1", "x2")

v4 <- as.data.frame(mvrnorm(n = 1000,
              mu = c(2,3),
              Sigma = matrix(c(1, -2, -2, 5), 2, 2, byrow = T) ) )
colnames(v4) = c("x1", "x2")
```

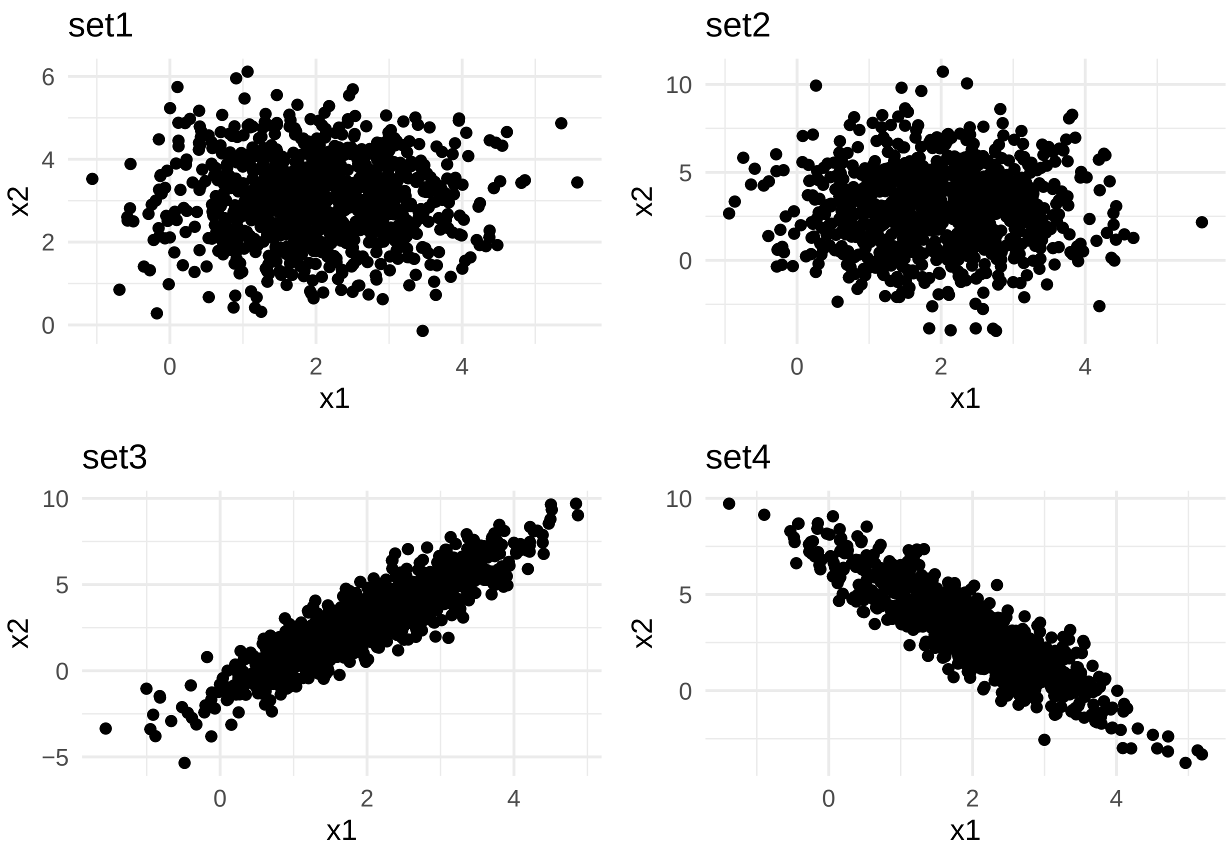
b) Plot the simulated distributions:

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
```

```
##
##     combine
```

```
p1 <-  ggplot(v1, aes(x1, x2)) + geom_point() + labs(title = "set1") + theme_minimal()
p2 <-  ggplot(v2, aes(x1, x2)) + geom_point() + labs(title = "set2") + theme_minimal()
p3 <-  ggplot(v3, aes(x1, x2)) + geom_point() + labs(title = "set3") + theme_minimal()
p4 <-  ggplot(v4, aes(x1, x2)) + geom_point() + labs(title = "set4") + theme_minimal()
grid.arrange(p1,p2,p3,p4, ncol = 2)
```



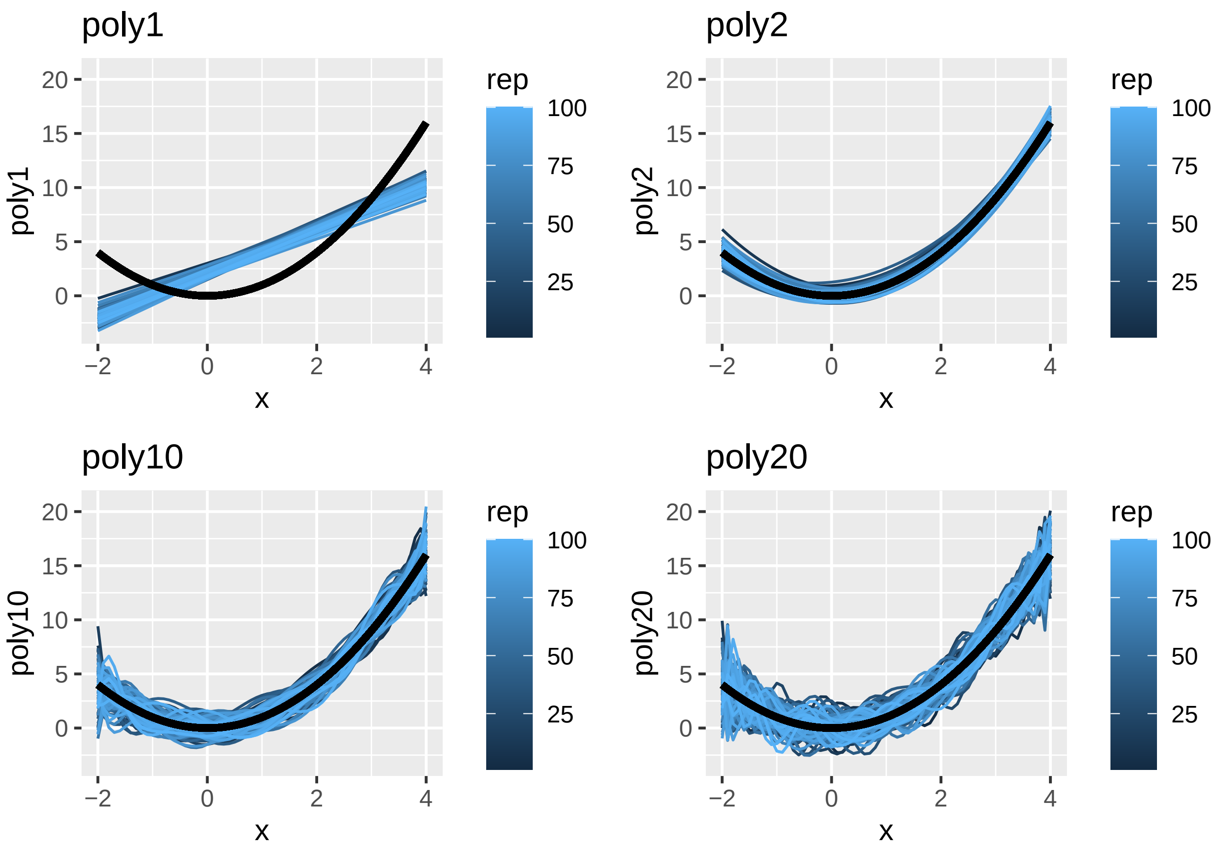
Problem 5

a) Supplied code:

```
library(ggplot2)
library(ggpubr)
set.seed(2)  # to reproduce
M = 100  # repeated samplings, x fixed
nord = 20  # order of polynoms
x = seq(from = -2, to = 4, by = 0.1)
truefunc = function(x) {
    return(x^2)
}
true_y = truefunc(x)
error = matrix(rnorm(length(x) * M, mean = 0, sd = 2), nrow = M, byrow = TRUE)
ymat = matrix(rep(true_y, M), byrow = T, nrow = M) + error
predarray = array(NA, dim = c(M, length(x), nord))
for (i in 1:M) {
    for (j in 1:nord) {
```

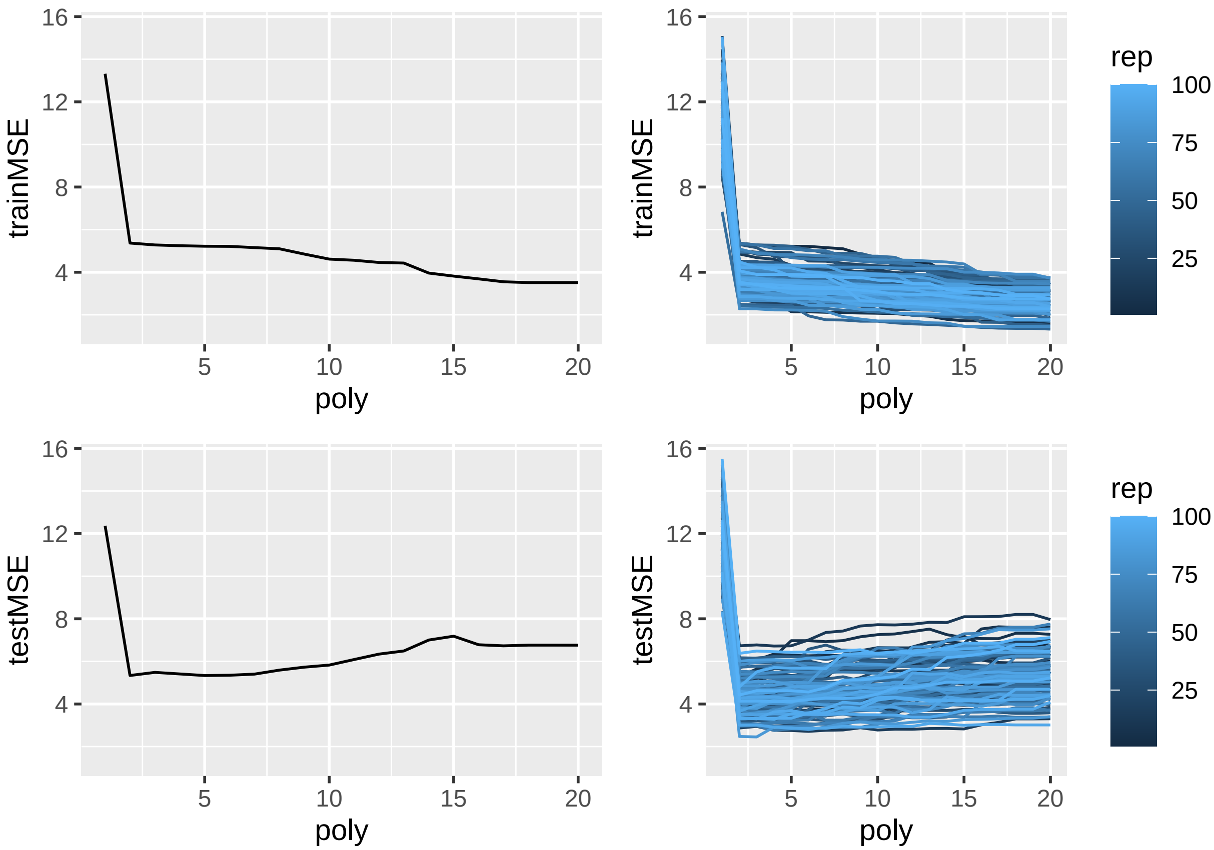
```r
        predarray[i, , j] = predict(lm(ymat[i, ] ~ poly(x, j, raw = TRUE)))
    }
}
# M matrices of size length(x) times nord first, only look at
# variablity in the M fits and plot M curves where we had 1 for
# plotting need to stack the matrices underneath eachother and make
# new variable 'rep'
stackmat = NULL
for (i in 1:M) {
    stackmat = rbind(stackmat, cbind(x, rep(i, length(x)), predarray[i,
        , ]))
}
# dim(stackmat)
colnames(stackmat) = c("x", "rep", paste("poly", 1:20, sep = ""))
sdf = as.data.frame(stackmat)  #NB have poly1-20 now - but first only use 1,2,20
# to add true curve using stat_function - easiest solution
true_x = x
yrange = range(apply(sdf, 2, range)[, 3:22])
p1 = ggplot(data = sdf, aes(x = x, y = poly1, group = rep, colour = rep)) +
    scale_y_continuous(limits = yrange) + geom_line()
p1 = p1 + stat_function(fun = truefunc, lwd = 1.3, colour = "black") +
    ggtitle("poly1")
p2 = ggplot(data = sdf, aes(x = x, y = poly2, group = rep, colour = rep)) +
    scale_y_continuous(limits = yrange) + geom_line()
p2 = p2 + stat_function(fun = truefunc, lwd = 1.3, colour = "black") +
    ggtitle("poly2")
p10 = ggplot(data = sdf, aes(x = x, y = poly10, group = rep, colour = rep)) +
    scale_y_continuous(limits = yrange) + geom_line()
p10 = p10 + stat_function(fun = truefunc, lwd = 1.3, colour = "black") +
    ggtitle("poly10")
p20 = ggplot(data = sdf, aes(x = x, y = poly20, group = rep, colour = rep)) +
    scale_y_continuous(limits = yrange) + geom_line()
p20 = p20 + stat_function(fun = truefunc, lwd = 1.3, colour = "black") +
    ggtitle("poly20")
ggarrange(p1, p2, p10, p20)
```

```
## Warning: Multiple drawing groups in `geom_function()`. Did you use the correct
## `group`, `colour`, or `fill` aesthetics?

## Warning: Multiple drawing groups in `geom_function()`. Did you use the correct
## `group`, `colour`, or `fill` aesthetics?

## Warning: Multiple drawing groups in `geom_function()`. Did you use the correct
## `group`, `colour`, or `fill` aesthetics?

## Warning: Multiple drawing groups in `geom_function()`. Did you use the correct
## `group`, `colour`, or `fill` aesthetics?
```

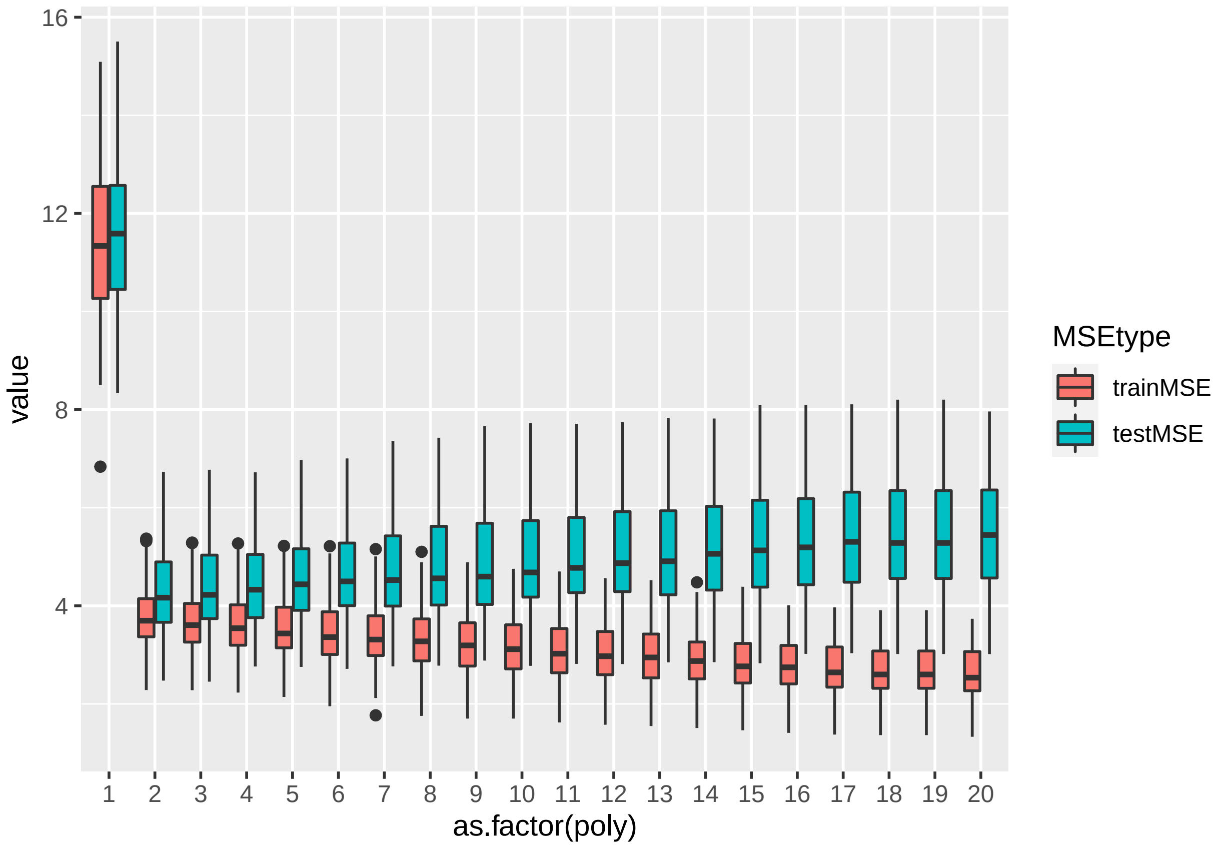


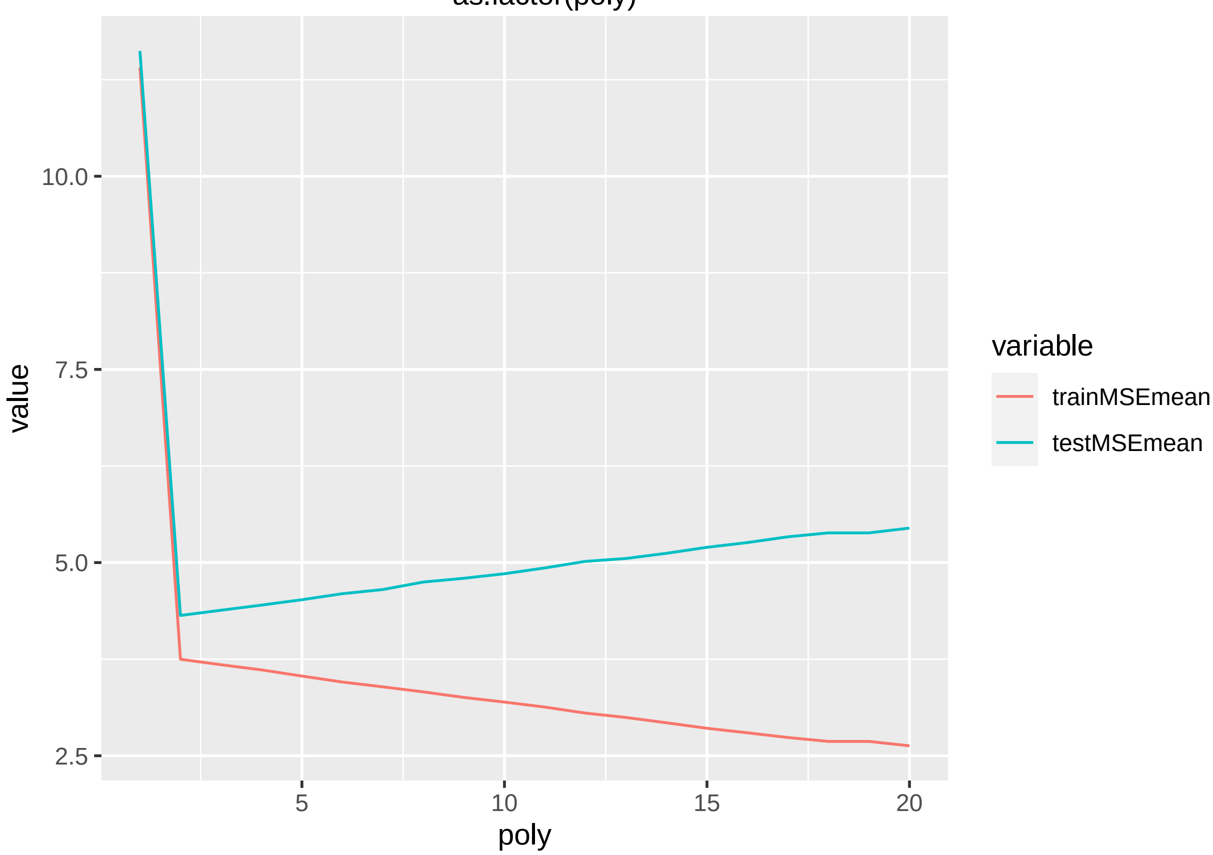
Unsuprisingly, the second degree polynomial fits best.

b) The code is found in on the exercise sheet, We get the following plots.



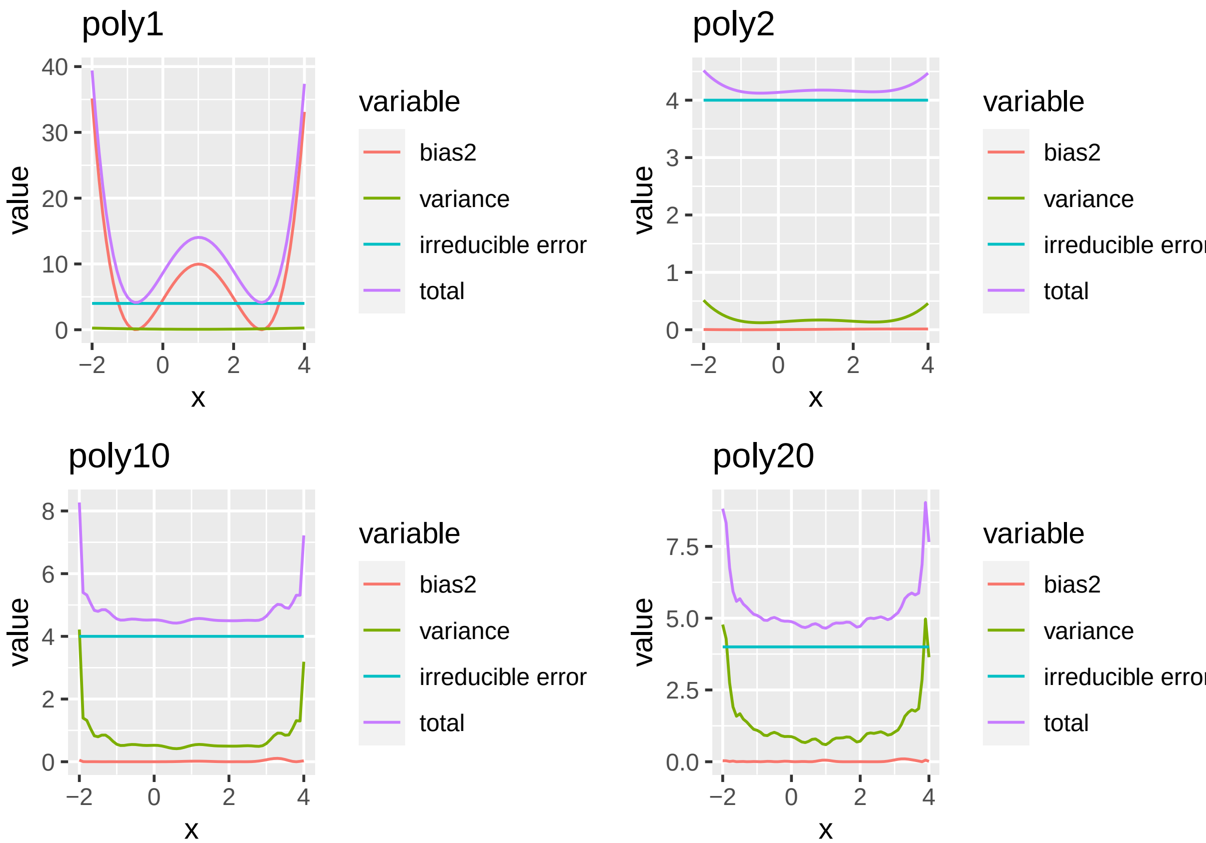

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

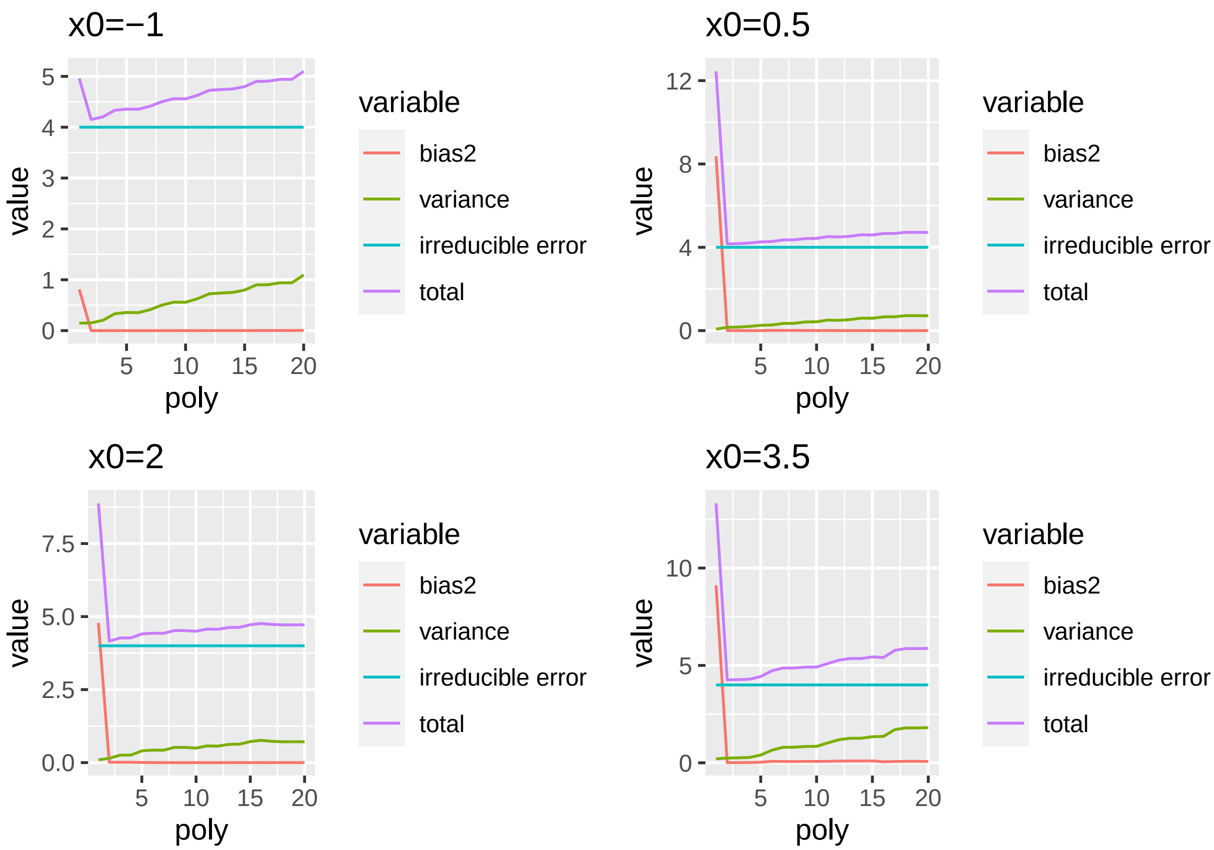




We can see that the second degreee polynimial achives the lowest men testMSE, while the average trainMSE increases with polynomial degree.

c)




x0=−1
value
poly
variable
bias2
variance
irreducible error
total
x0=0.5
value
poly
variable
bias2
variance
irreducible error
total
x0=2
value
poly
variable
bias2
variance
irreducible error
total
x0=3.5
value
poly
variable
bias2
variance
irreducible error
total

d)