HW 5

Q1

1.1

```r
#rm(list=ls())
csv1 <- read.csv('model1.csv')
summary(lm(y ~ d + x1, data = csv1))
```

```
##
## Call:
## lm(formula = y ~ d + x1, data = csv1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6680 -0.2548 -0.2548  0.3320  0.9564
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.043571   0.005051   8.626   <2e-16 ***
## d           0.413189   0.005266  78.466   <2e-16 ***
## x1          0.211207   0.005127  41.191   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4321 on 99997 degrees of freedom
## Multiple R-squared:  0.05828,    Adjusted R-squared:  0.05826
## F-statistic:  3094 on 2 and 99997 DF,  p-value: < 2.2e-16
```

ATE is 0.413. We want to adjust for non-causal correlation, X1, but not block causal paths through X2.

1.2

```r
csv2 <- read.csv('model2.csv')
summary(lm(y ~ d + x1 + x2, data = csv2))
```

```
##
## Call:
## lm(formula = y ~ d + x1 + x2, data = csv2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7527 -0.2126 -0.1937  0.2473  1.0669
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.066891   0.001959  -34.14   <2e-16 ***
## d            0.260606   0.003045   85.60   <2e-16 ***
```

```
## x1           0.279484   0.002508  111.45   <2e-16 ***
## x2           0.279500   0.002518  111.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3536 on 99996 degrees of freedom
## Multiple R-squared:  0.3915, Adjusted R-squared:  0.3914
## F-statistic: 2.144e+04 on 3 and 99996 DF,  p-value: < 2.2e-16
```

ATE is 0.260606. We want to adjust for non-causal correlation.

1.3

```r
csv3 <- read.csv('model3.csv')
summary(lm(y ~ d + x1, data = csv3))
```

```
##
## Call:
## lm(formula = y ~ d + x1, data = csv3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7057 -0.5022  0.2943  0.3561  0.5596
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.440410   0.004020  109.56   <2e-16 ***
## d           0.061789   0.003266   18.92   <2e-16 ***
## x1          0.203493   0.003533   57.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.476 on 99997 degrees of freedom
## Multiple R-squared:  0.03213,    Adjusted R-squared:  0.03211
## F-statistic:  1660 on 2 and 99997 DF,  p-value: < 2.2e-16
```

ATE is 0.061789. We want to adjust for unknown confounders.

Q2

2.1

We do not know U – an unknown confounder. So, we cannot identify ATE via simple regression adjustment.

2.2

$Y_i = \lambda_{dy} D_i + U_i = \lambda_{dy}\lambda_{xd}U_{xi} + \lambda_{dy}U_i + U_i$. The bias is obviously larger when we have X adjusted: for $Y_i = \lambda_{dy}D_i + kX_i + U_i = \lambda_{dy}\lambda_{xd}U_{xi} + \lambda_{dy}U_i + U_i + k\lambda_{xd}U_{xi} + kU_i$. Therefore, there is a bias amplification effect.

2.3

$Cov(X, Y) = Cov(X, \lambda_{dy}D_i + U_i) = Cov(X, \lambda_{dy}D_i) = \lambda_{dy}Cov(X, D_i)$

Thus, $\lambda_{dy} = \frac{Cov(X,Y)}{Cov(X,D_i)} = \frac{\frac{Cov(X,Y)}{Var(X)}}{\frac{Cov(X,D_i)}{Var(X)}} = \frac{\beta_{yx}}{\beta_{dx}}$.

Q3

3.1

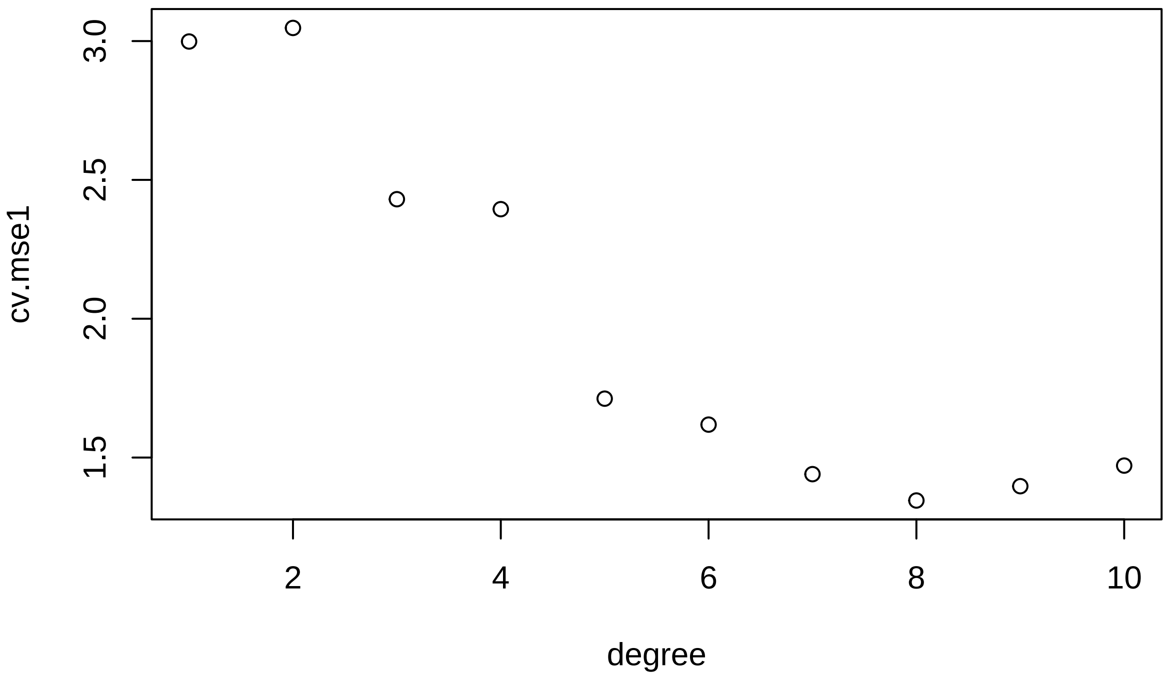
```r
# function to compute k-fold CV MSE for polynomial regression
cv_mse <- function(data, d = 1, k = 10){

    # create folds randomly
    n <- nrow(data)
    folds <- sample(rep(1:k, length = n))

    # create vector to store results
    mse <- rep(NA, k)
    for(j in 1:k){

        # train model on all folds except j
        train <- folds != j
        ols  <- lm(y ~ poly(x, degree = d, raw = T), data = data[train, ])

        # compute MSE on fold j (not used for training)
        yhat <- predict(ols, newdata = data[!train, ])
        mse[j]  <- mean((data$y[!train] - yhat)^2)
  }
    # compute average mse
    mse.cv <- mean(mse)
    return(mse.cv)
}
```

Function

```r
set.seed(42)
n = 100
x = runif(n, -4, 4)
e = rnorm(n)

#?ifelse

# simulate DGP1
y  <- -2*ifelse(x < -3, 1,0) + 2.55 * ifelse(x > -2, 1,0) - 2*ifelse(x > 0, 1,0) +
    4 * ifelse(x > 2, 1,0) - ifelse(x > 3, 1,0) + e
sim_data1 <- data.frame(x, y)

# simulate DGP2
y  <- 6 + 0.4 * x - 0.36 * x^2 + 0.006 * x^3 + e
sim_data2 <- data.frame(x, y)

# simulate DGP3
y  <- 2.83*sin(pi/2*x) + e
sim_data3 <- data.frame(x, y)

# simulate DGP4
y  <- 4*sin(3*pi *x) * ifelse(x > 0, 1,0) + e
sim_data4 <- data.frame(x, y)
```

DGP

```r
# plot size
options(repr.plot.width = 10, repr.plot.height = 10)

# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse1  <- sapply(degree, function(d) cv_mse(sim_data1, d))
plot(degree,cv.mse1)
```



Sim data 1

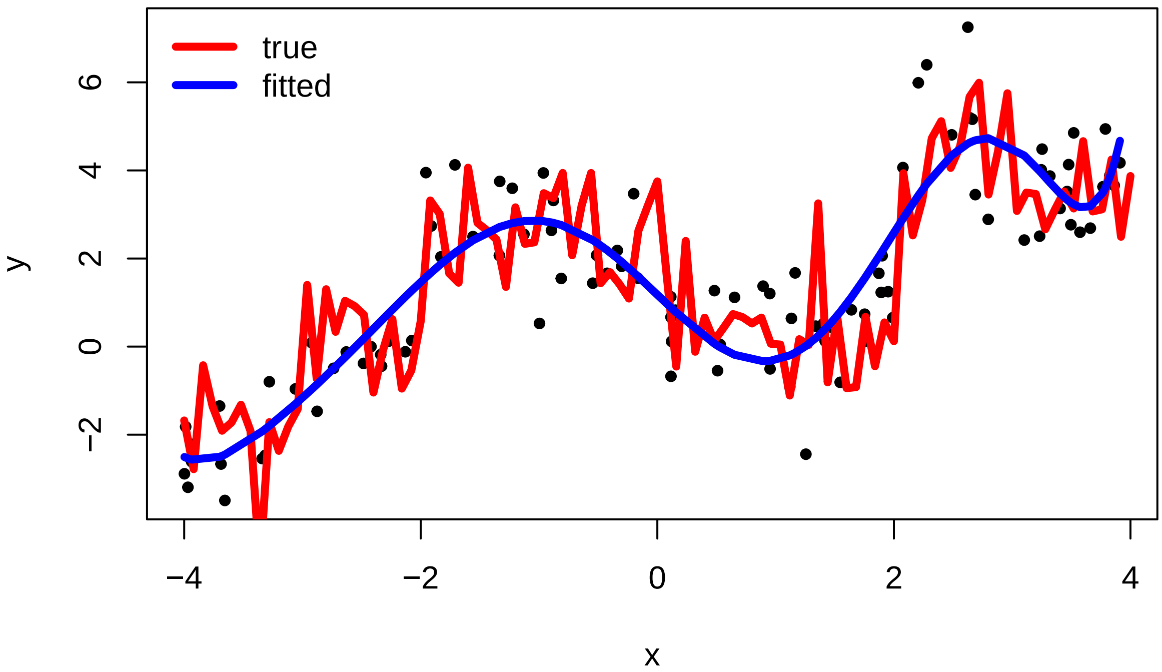
```r
# best degree
best1 <- degree[which.min(cv.mse1)]
best1
```

```
## [1] 8
```

```r
# fit model using best degree
ols1 <- lm(y ~ poly(x, degree = best1, raw = T), data = sim_data1)

# predicted values
yhat1 <- predict(ols1)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data1, pch= 20)
curve( -2*ifelse(x < -3, 1,0) + 2.55 * ifelse(x > -2, 1,0) - 2*ifelse(x > 0, 1,0)
       + 4 * ifelse(x > 2, 1,0) - ifelse(x > 3, 1,0) + e, col = "red", from = -4, to = 4, add = T, lwd =
```

```
## Warning in -2 * ifelse(x < -3, 1, 0) + 2.55 * ifelse(x > -2, 1, 0) - 2 * :
## longer object length is not a multiple of shorter object length
```

```r
lines(yhat1[order(x)] ~ sort(x), data= sim_data1, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```

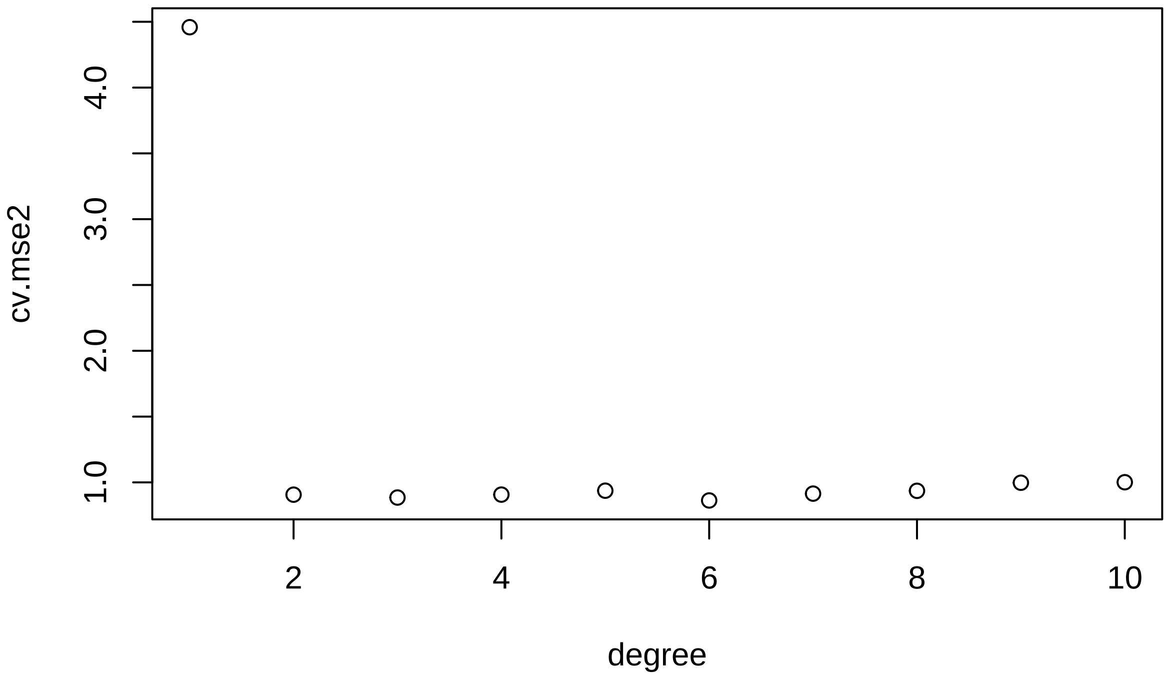



```
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse2 <- sapply(degree, function(d) cv_mse(sim_data2, d))

# best degree
best2 <- degree[which.min(cv.mse2)]
best2
```

Sim data 2

```
## [1] 6
```

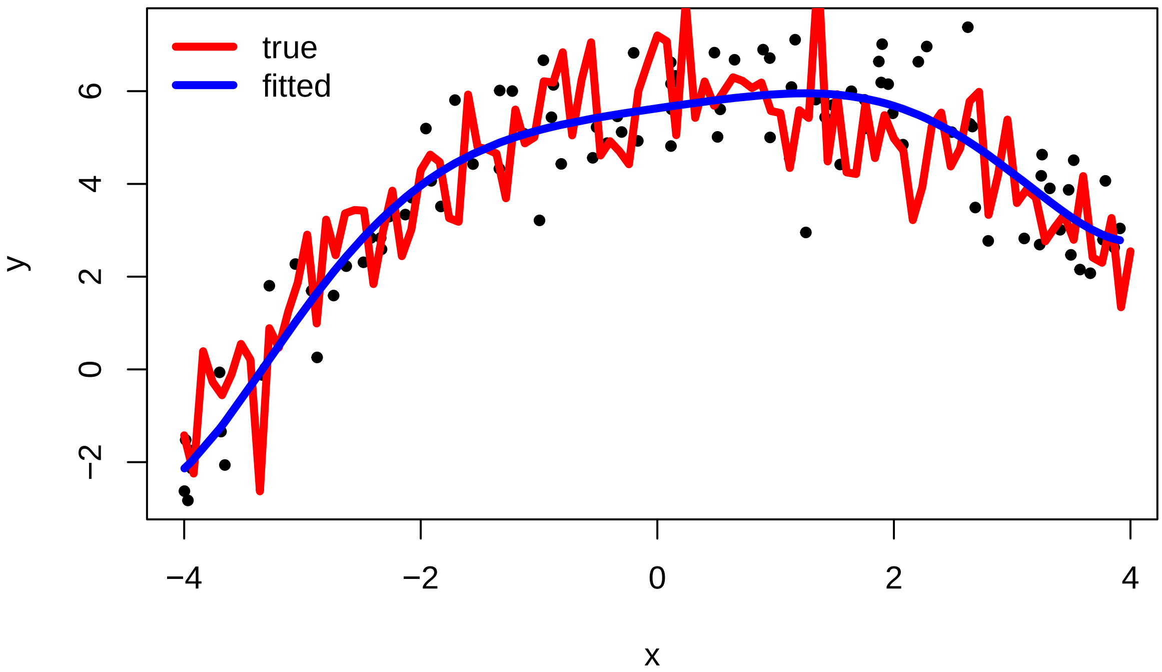
```
plot(degree,cv.mse2)
```




```
# fit model using best degree
ols2 <- lm(y ~ poly(x, degree = best2, raw = T), data = sim_data2)

# predicted values
yhat2 <- predict(ols2)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data2, pch= 20)
curve( 6 + 0.4 * x - 0.36 * x^2 + 0.006 * x^3 + e, col = "red", from = -4, to = 4, add = T, lwd = 4)
```

```
## Warning in 6 + 0.4 * x - 0.36 * x^2 + 0.006 * x^3 + e: longer object length is
## not a multiple of shorter object length
```

```
lines(yhat2[order(x)] ~ sort(x), data= sim_data2, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```

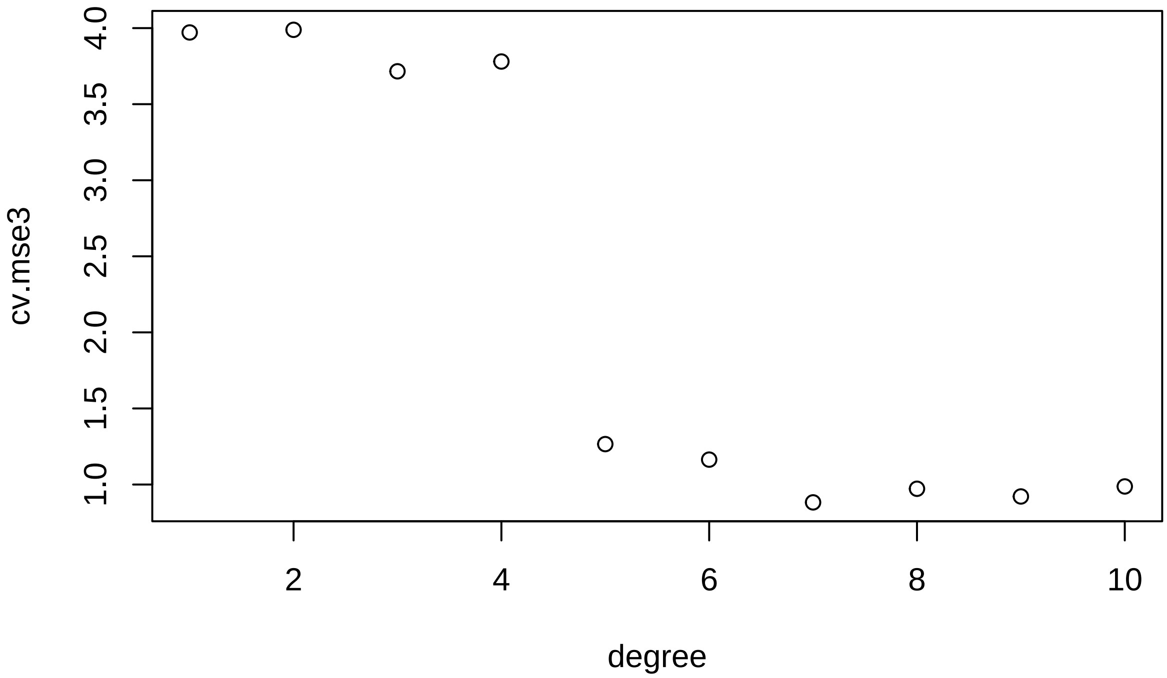



```
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse3 <- sapply(degree, function(d) cv_mse(sim_data3, d))

# best degree
best3 <- degree[which.min(cv.mse3)]
best3
```

Sim data 3

```
## [1] 7
```

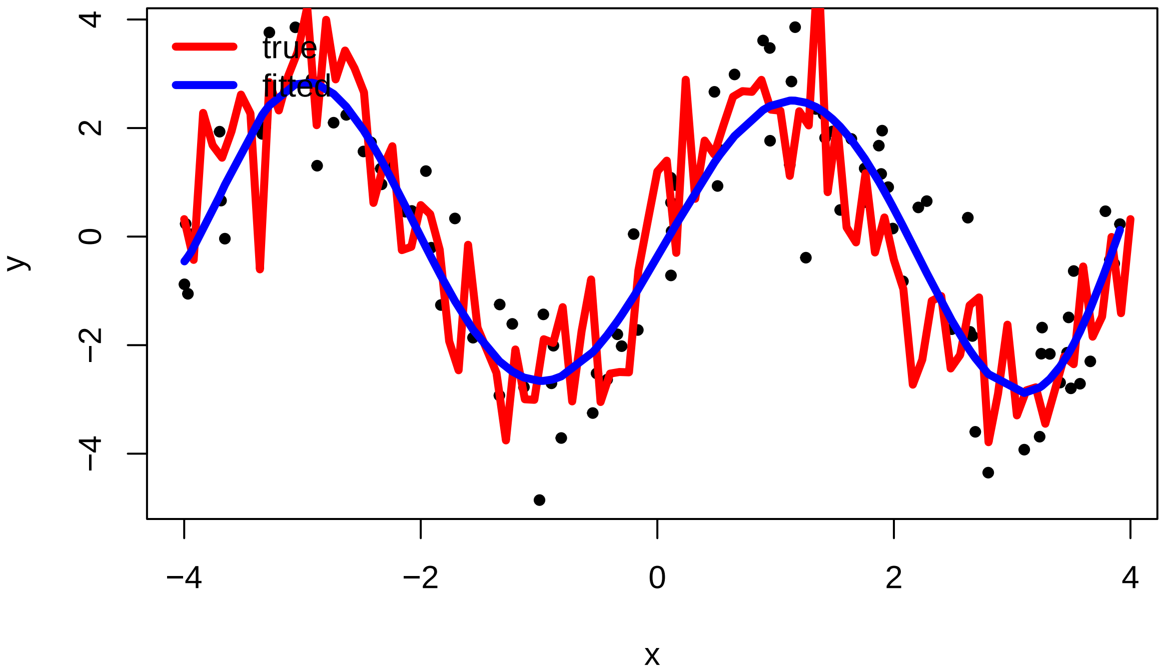
```
plot(degree,cv.mse3)
```




```
# fit model using best degree
ols3 <- lm(y ~ poly(x, degree = best3, raw = T), data = sim_data3)

# predicted values
yhat3 <- predict(ols3)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data3, pch= 20)
curve(2.83*sin(pi/2*x) + e, col = "red", from = -4, to = 4, add = T, lwd = 4)
```

```
## Warning in 2.83 * sin(pi/2 * x) + e: longer object length is not a multiple of
## shorter object length
```

```
lines(yhat3[order(x)] ~ sort(x), data= sim_data3, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```



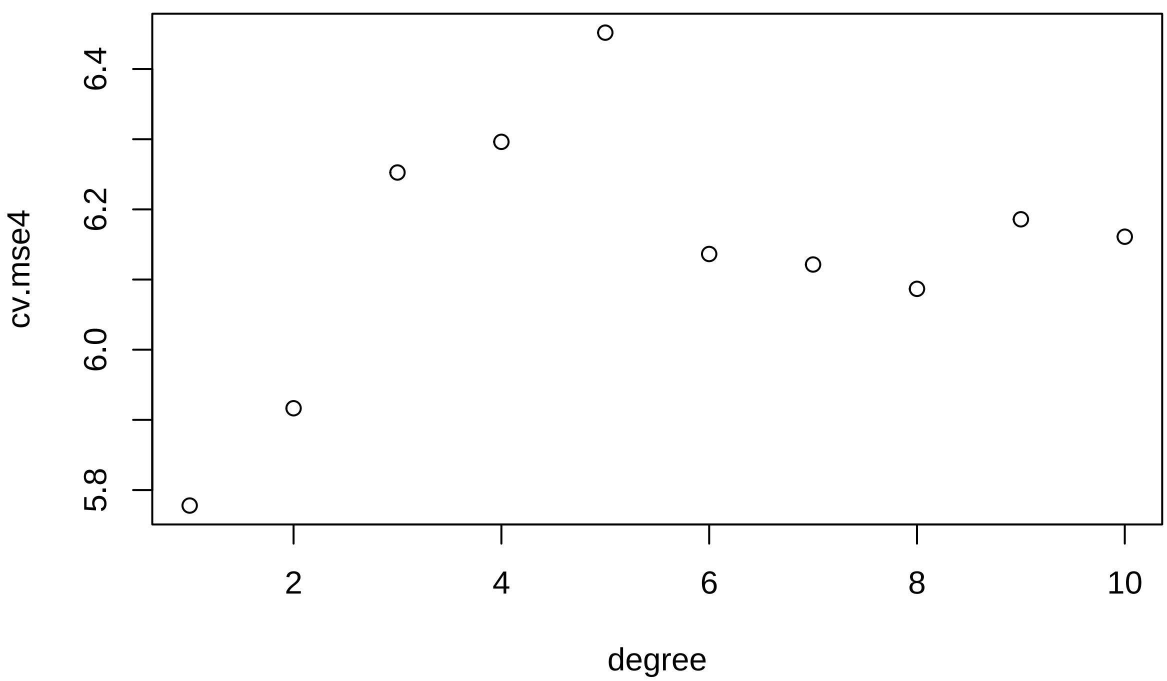
####

Sim data 4

```r
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse4 <- sapply(degree, function(d) cv_mse(sim_data4, d))

# best degree
best4 <- degree[which.min(cv.mse4)]
best4
```

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

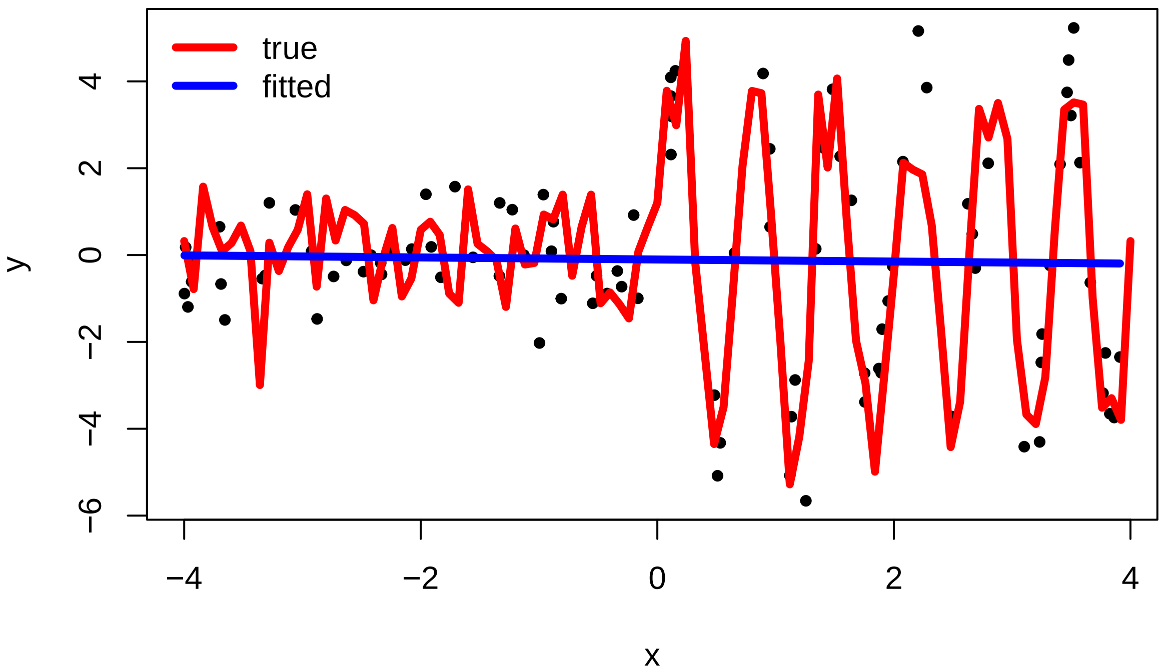
```r
plot(degree,cv.mse4)
```



```r
# fit model using best degree
ols4 <- lm(y ~ poly(x, degree = best4, raw = T), data = sim_data4)

# predicted values
yhat4 <- predict(ols4)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data4, pch= 20)
curve(4*sin(3*pi *x) * ifelse(x > 0, 1,0) + e, col = "red", from = -4, to = 4, add = T, lwd = 4)
```

```
## Warning in 4 * sin(3 * pi * x) * ifelse(x > 0, 1, 0) + e: longer object length
## is not a multiple of shorter object length
```

```r
lines(yhat4[order(x)] ~ sort(x), data= sim_data4, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```



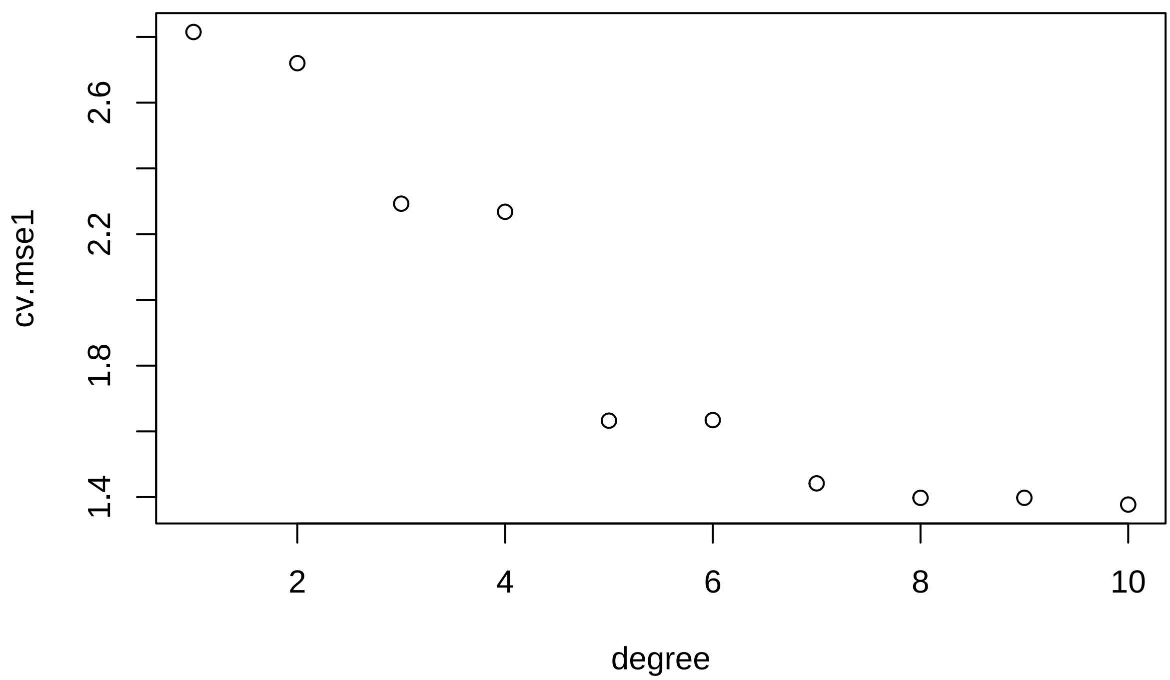
3.2

```r
set.seed(42)
n = 1000
x = runif(n, -4, 4)
e = rnorm(n)

#?ifelse

# simulate DGP1
y  <- -2*ifelse(x < -3, 1,0) + 2.55 * ifelse(x > -2, 1,0) - 2*ifelse(x > 0, 1,0) +
    4 * ifelse(x > 2, 1,0) - ifelse(x > 3, 1,0) + e
sim_data1 <- data.frame(x, y)

# simulate DGP2
```

```r
y  <- 6 + 0.4 * x - 0.36 * x^2 + 0.006 * x^3 + e
sim_data2 <- data.frame(x, y)

# simulate DGP3
y  <- 2.83*sin(pi/2*x) + e
sim_data3 <- data.frame(x, y)

# simulate DGP4
y  <- 4*sin(3*pi *x) * ifelse(x > 0, 1,0) + e
sim_data4 <- data.frame(x, y)
```

DGP

```r
# plot size
options(repr.plot.width = 10, repr.plot.height = 10)

# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse1  <- sapply(degree, function(d) cv_mse(sim_data1, d))
plot(degree,cv.mse1)
```



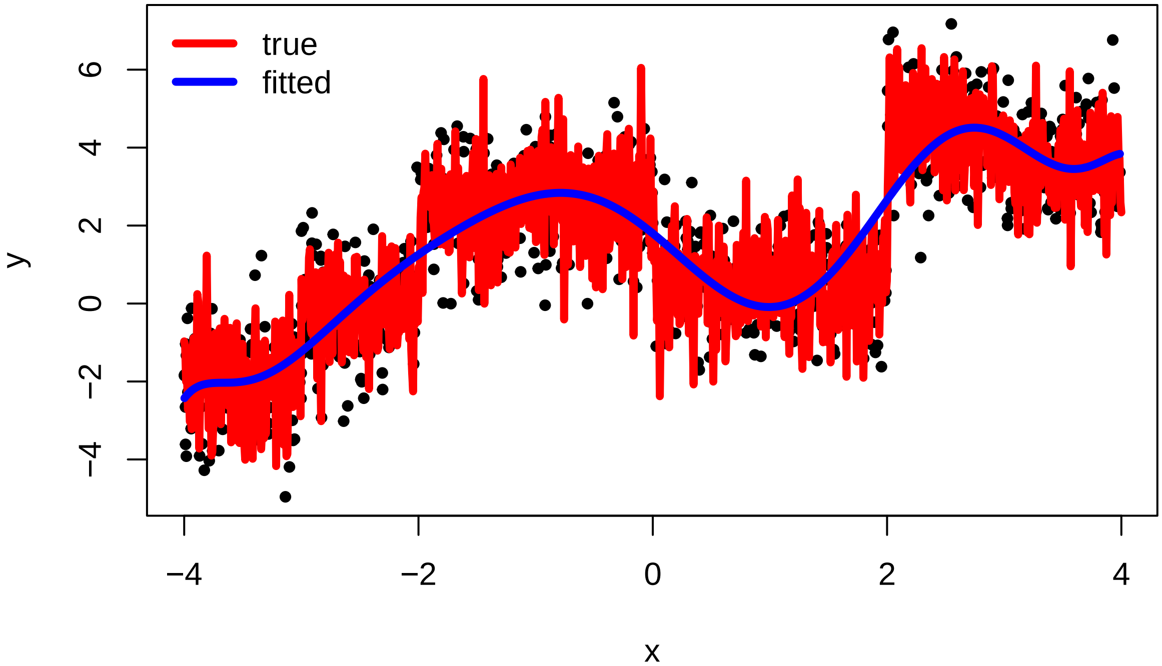
Sim data 1

```r
# best degree
best1 <- degree[which.min(cv.mse1)]
best1
```

```
## [1] 10
```

```r
# fit model using best degree
ols1 <- lm(y ~ poly(x, degree = best1, raw = T), data = sim_data1)

# predicted values
yhat1 <- predict(ols1)
```

```
# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data1, pch= 20)
curve( -2*ifelse(x < -3, 1,0) + 2.55 * ifelse(x > -2, 1,0) - 2*ifelse(x > 0, 1,0)+
         4 * ifelse(x > 2, 1,0) - ifelse(x > 3, 1,0) + e, col = "red", from = -4, to = 4, n =n, add = T
lines(yhat1[order(x)] ~ sort(x), data= sim_data1, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```

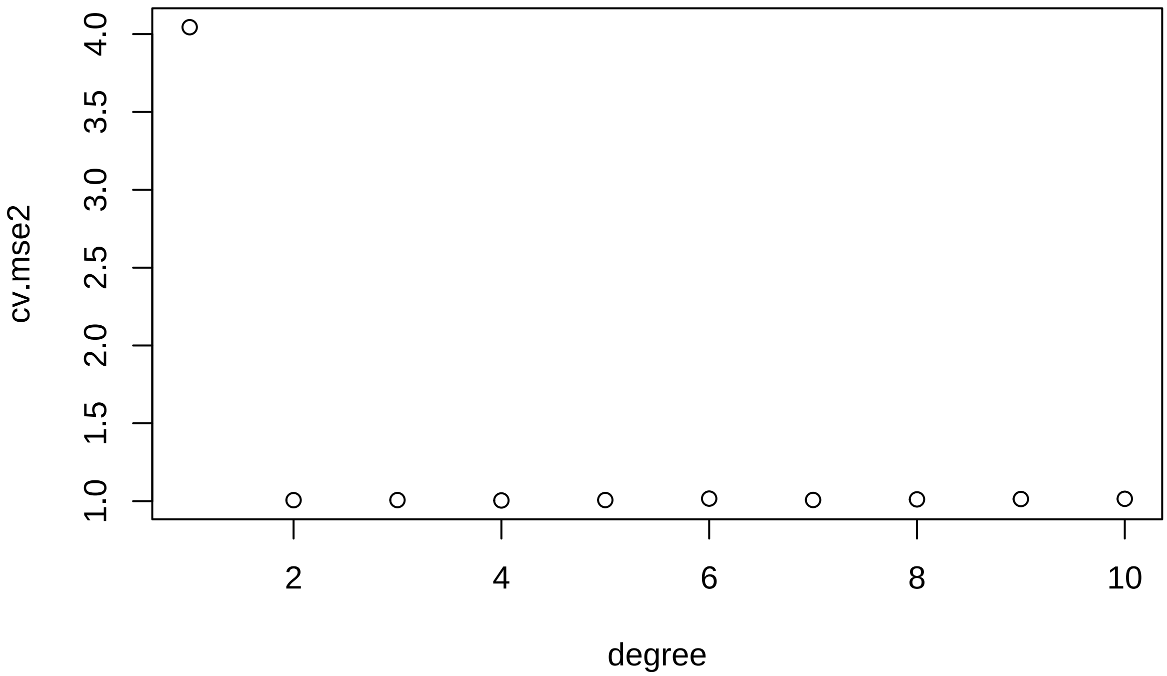



```
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse2 <- sapply(degree, function(d) cv_mse(sim_data2, d))

# best degree
best2 <- degree[which.min(cv.mse2)]
best2
```

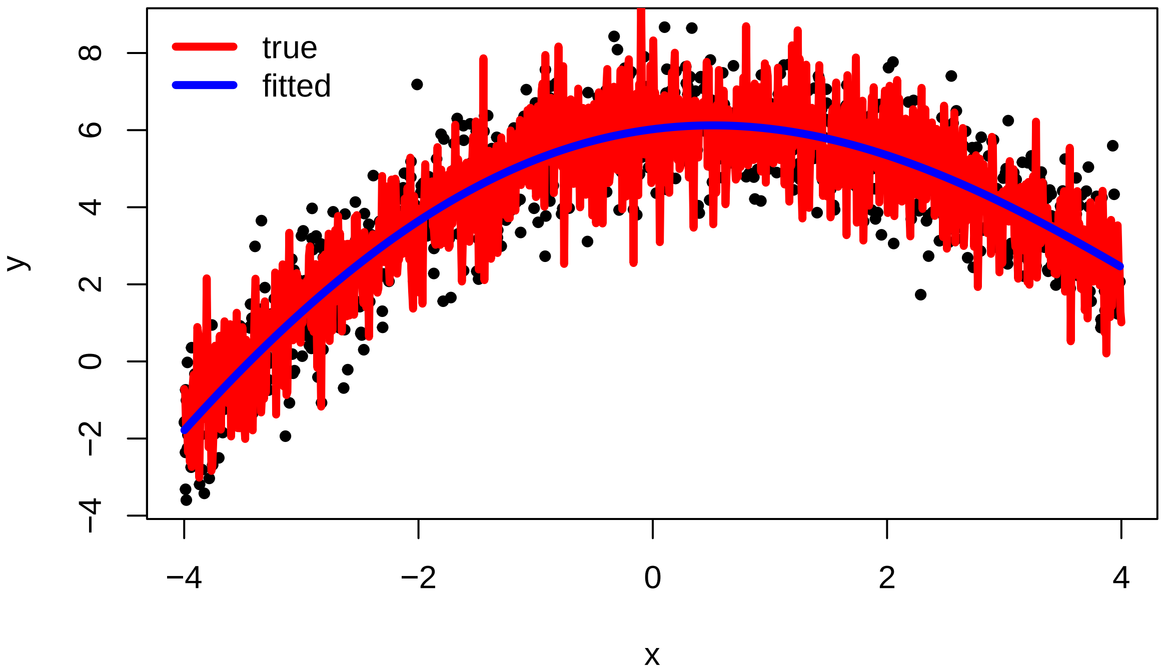
Sim data 2

```
## [1] 4
```

```
plot(degree,cv.mse2)
```




```r
# fit model using best degree
ols2 <- lm(y ~ poly(x, degree = best2, raw = T), data = sim_data2)

# predicted values
yhat2 <- predict(ols2)
?curve

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data2, pch= 20)
curve( 6 + 0.4 * x - 0.36 * x^2 + 0.006 * x^3 + e, col = "red", from = -4, to = 4, n =n, add = T, lwd =
lines(yhat2[order(x)] ~ sort(x), data= sim_data2, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```

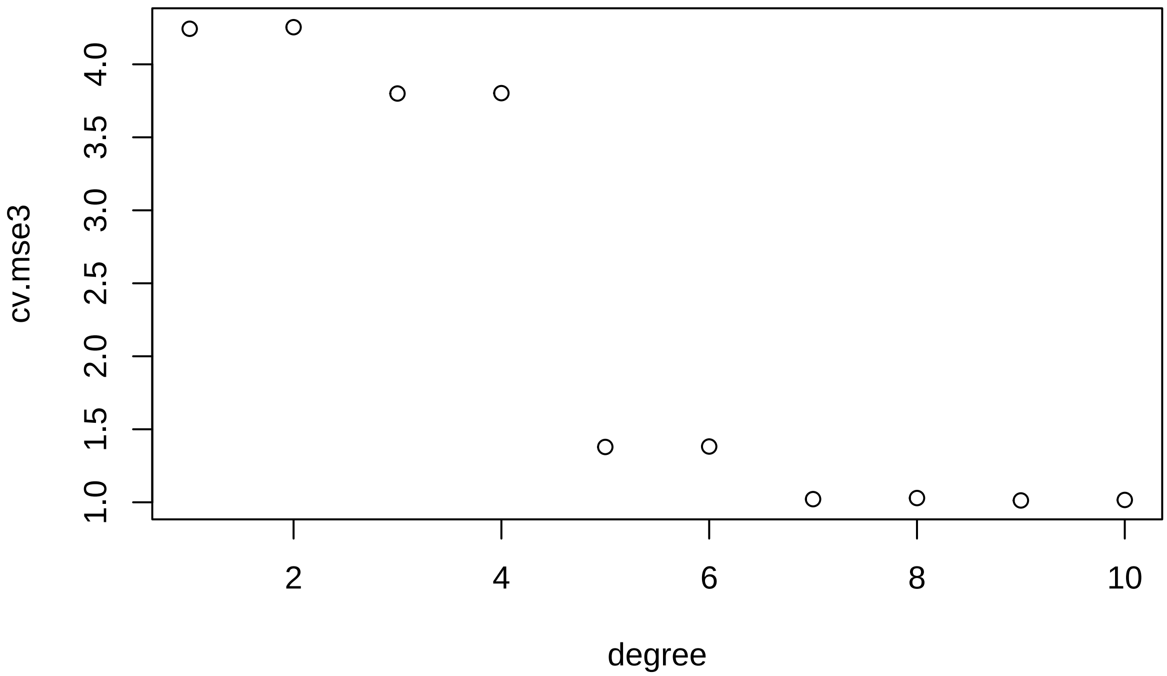



```
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse3 <- sapply(degree, function(d) cv_mse(sim_data3, d))

# best degree
best3 <- degree[which.min(cv.mse3)]
best3
```

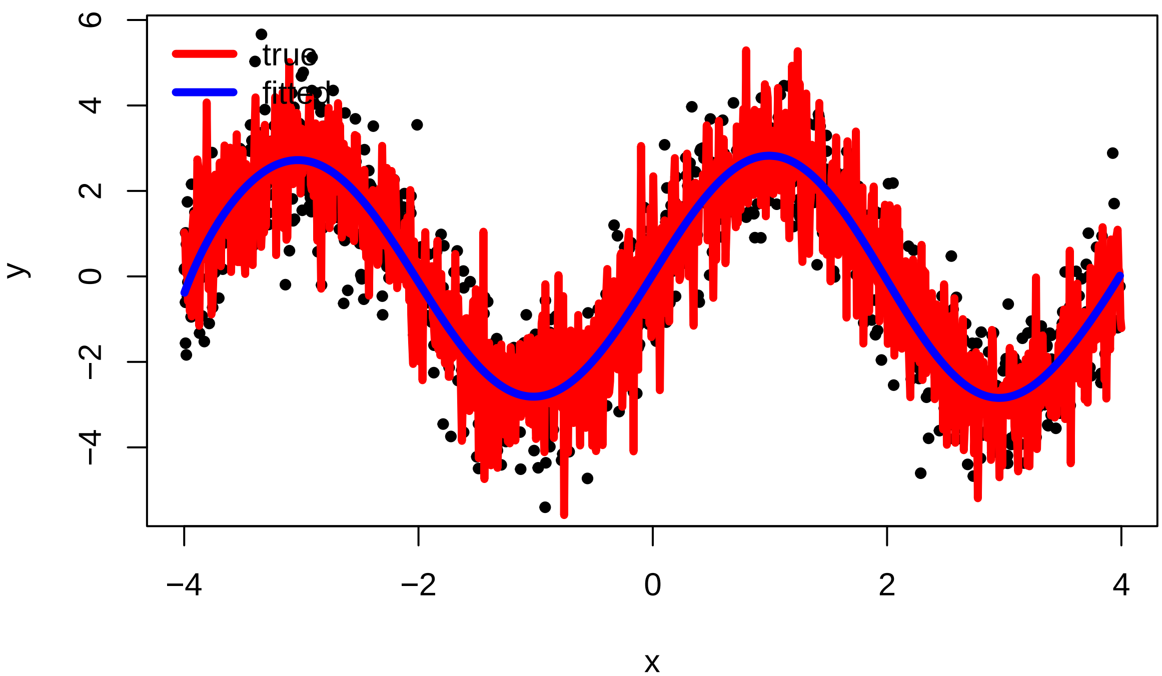
Sim data 3

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

```
plot(degree,cv.mse3)
```




```
# fit model using best degree
ols3 <- lm(y ~ poly(x, degree = best3, raw = T), data = sim_data3)

# predicted values
yhat3 <- predict(ols3)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data3, pch= 20)
curve(2.83*sin(pi/2*x) + e, col = "red", from = -4, to = 4, n=n, add = T, lwd = 4)
lines(yhat3[order(x)] ~ sort(x), data= sim_data3, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```

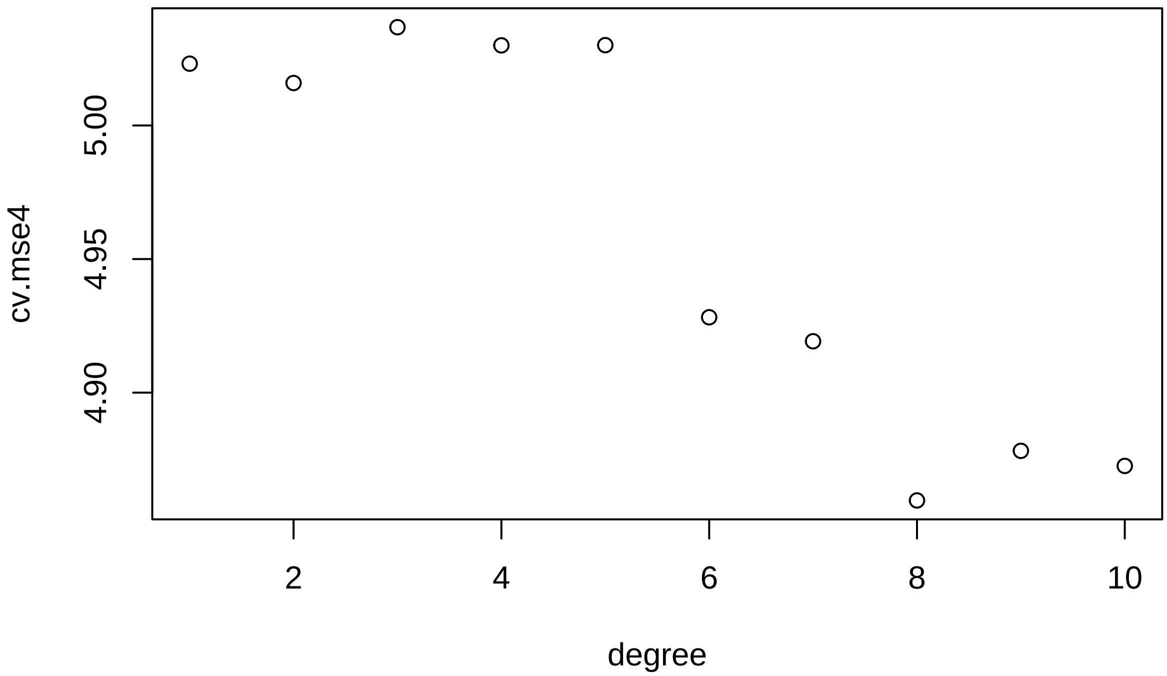



```
# compute MSE's for d from 1 to 10
degree <- 1:10
cv.mse4 <- sapply(degree, function(d) cv_mse(sim_data4, d))

# best degree
best4 <- degree[which.min(cv.mse4)]
best4
```

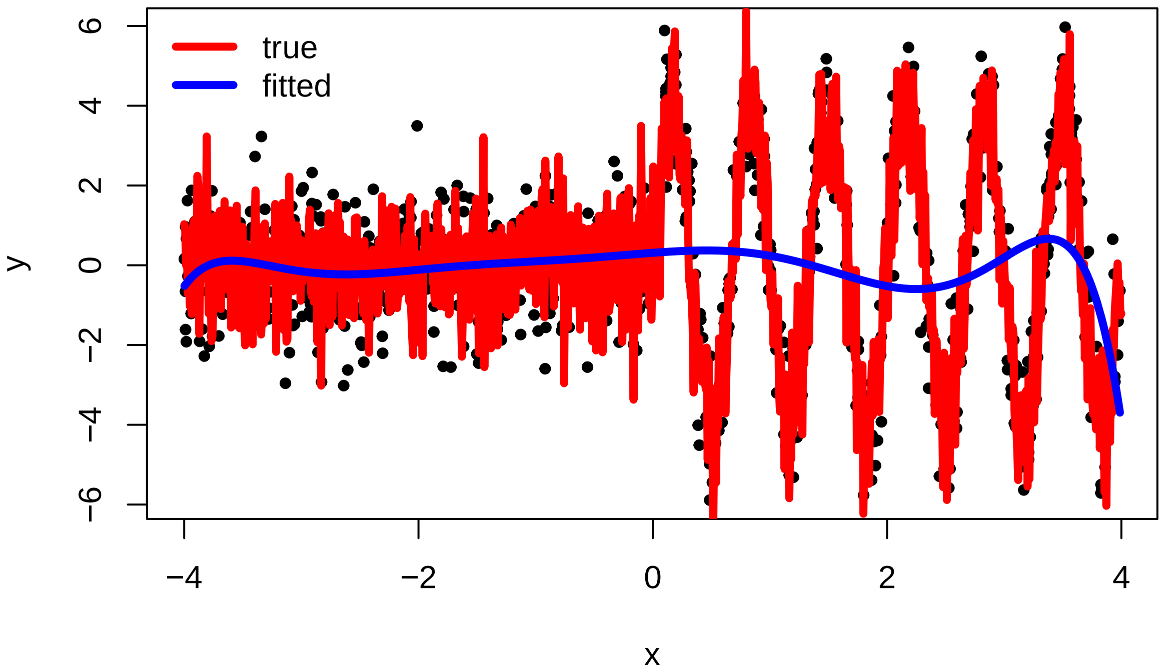
Sim data 4

```
## [1] 8
```

```
plot(degree,cv.mse4)
```




```
# fit model using best degree
ols4 <- lm(y ~ poly(x, degree = best4, raw = T), data = sim_data4)

# predicted values
yhat4 <- predict(ols4)

# plot against data and true CEF
plot.new()
plot(y ~ x, sim_data4, pch= 20)
curve(4*sin(3*pi *x) * ifelse(x > 0, 1,0) + e, col = "red", n=n, from = -4, to = 4, add = T, lwd = 4)
lines(yhat4[order(x)] ~ sort(x), data= sim_data4, col = "blue", lwd = 4)
legend("topleft", col = c("red", "blue"), legend = c("true", "fitted"), lty = 1, bty = "n", lwd = 4)
```



The results are different because more data can better help us fit the models.

Q4

4.1

```
tecator <- read.csv('tecator.csv')
ols_tecator <- lm(fat~ ., data= tecator)
mean((predict(ols_tecator) - tecator$fat)^2)
```

```
## [1] 0.7898249
```

This possibly is not a good estimate of out-of-sample performance because there are too many predictors.

4.2

```
cv_mse1 <- function(data, k = 5){
    set.seed(59)
    # create folds randomly
    n <- nrow(data)
    folds <- sample(rep(1:k, length = n))

    # create vector to store results
    mse <- rep(NA, k)
    for(j in 1:k){

        # train model on all folds except j
        train <- folds != j
        ols  <- lm(fat ~ ., data = data[train, ])

        # compute MSE on fold j (not used for training)
        yhat <- predict(ols, newdata = data[!train, ])
        mse[j]  <- mean((data$fat[!train] - yhat)^2)
```

```
  }
    # compute average mse
    mse.cv <- mean(mse)
    return(mse.cv)
}
cv_mse1(tecator)
```

```
## [1] 13.60371
```

The MSE is different from the OLS model, and interestingly larger. Possibly due to the variation in predictors.

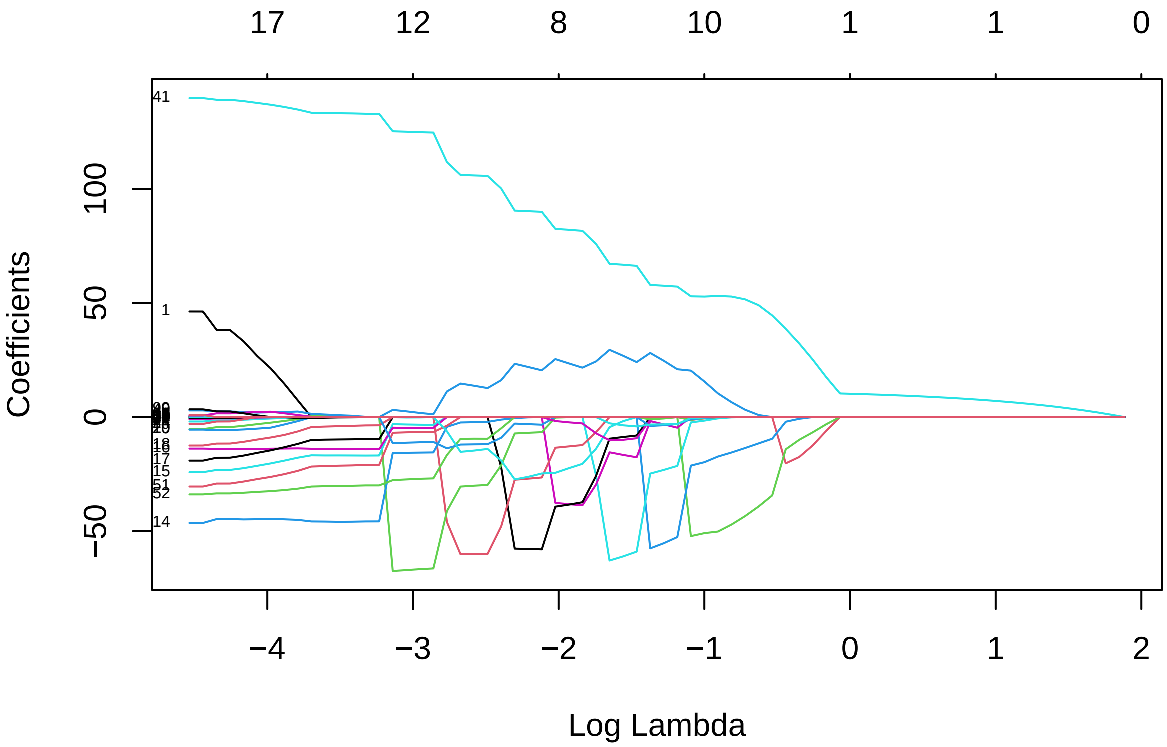
4.3

```
# load package
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

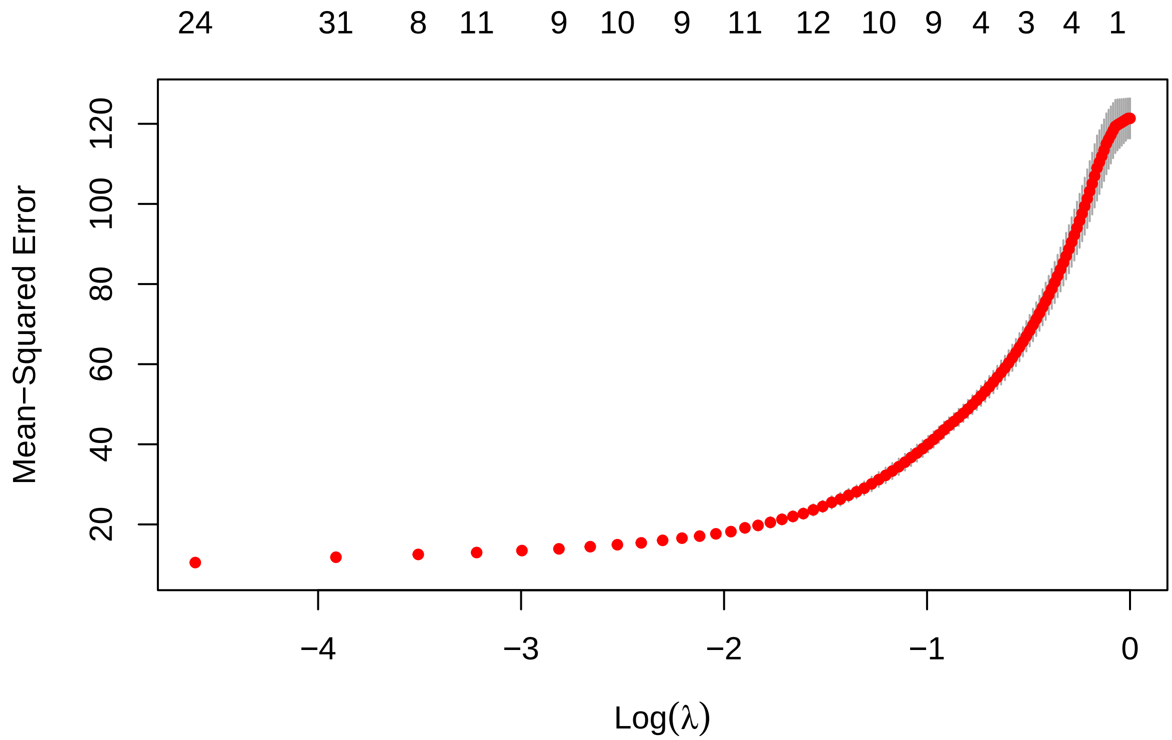
```
# plot size
options(repr.plot.width = 10, repr.plot.height = 10)

# scipen
options(scipen = 99)

# create model matrix
X <- model.matrix(fat ~ . -1, data = tecator)
y <- tecator$fat

# alpha = 1 (Lasso)
lasso <- glmnet(X, y, alpha = 1)
plot(lasso, xvar = "lambda", label = T)
```




```
cv.lasso <- cv.glmnet(X, y, type = "mse", nfolds = 5, nlambda = 100, lambda = seq(0, 1, by = 0.01), alph

cv.lasso
```

```
##
## Call:  cv.glmnet(x = X, y = y, lambda = seq(0, 1, by = 0.01), type.measure = "mse",      nfolds = 5,
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min      0   101   9.216 0.9126     100
## 1se      0   101   9.216 0.9126     100
```

```
plot(cv.lasso)
```




```
coef(cv.lasso, s = "lambda.min")
```

```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)  12.33362263711
## X1           26.25016782401
## X2            0.00132744517
## X3            0.00128815092
## X4          -17.14430935904
## X5           24.31156909980
## X6           22.58445823756
## X7           20.97362218397
## X8           20.68858272735
## X9           15.94540874881
## X10         -49.99717113627
## X11         -62.24684034308
## X12         -26.70717792041
## X13         220.84369109788
## X14          -1.14726566823
## X15         -55.65582735566
## X16         -42.25910340554
## X17         -36.23956649454
## X18         -33.22480557033
## X19         -27.90113824952
## X20         -25.14916900822
## X21         -20.87778344594
## X22         -15.80940554138
## X23          -8.59676177743
## X24           1.93375256439
## X25          10.21005429332
## X26          12.18680783500
```

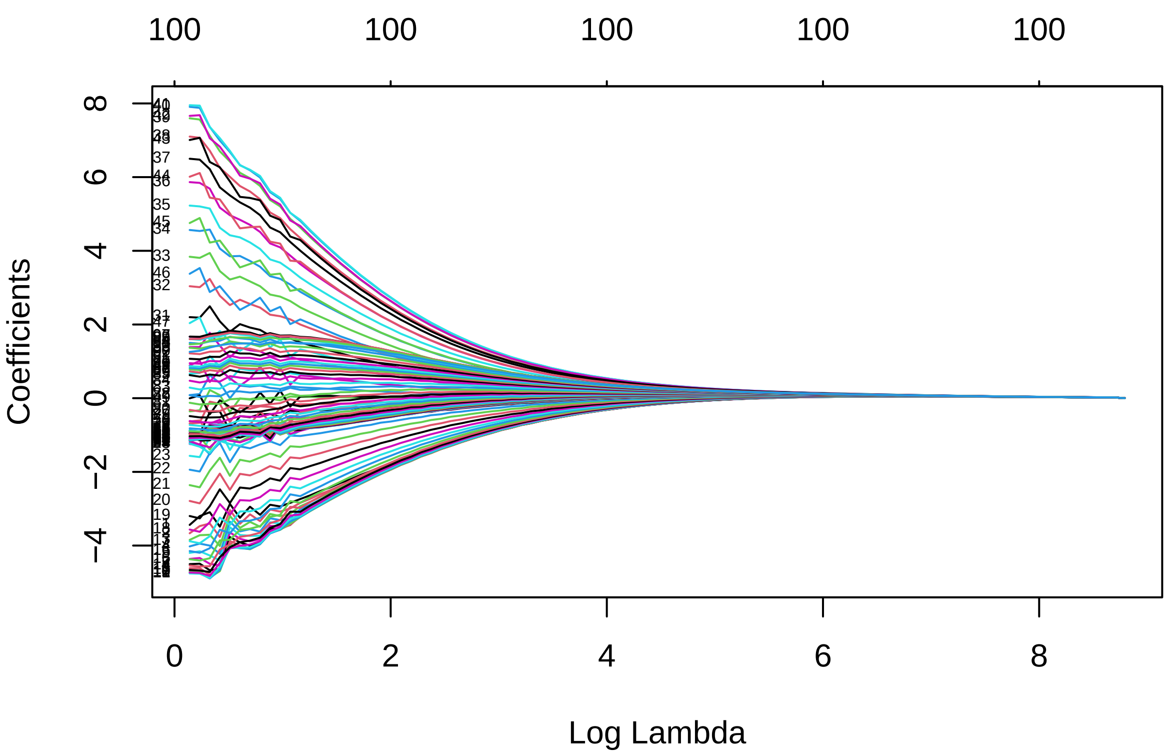
```
## X27           8.52427175847
## X28           2.25622547849
## X29          -2.05424550914
## X30          -8.82270298516
## X31         -17.27352129986
## X32         -22.41790490740
## X33         -21.05215769630
## X34         -11.45502422799
## X35          -0.06353441172
## X36           9.43569470116
## X37          15.23284643548
## X38          16.60688789648
## X39          10.64857935571
## X40           8.74178860766
## X41         141.38501824871
## X42           0.00013796077
## X43           0.00024574255
## X44           0.00006437054
## X45          -0.00029220556
## X46          -0.00064655588
## X47          -0.00076993003
## X48          -0.00063047886
## X49           0.00023394090
## X50         -31.41070306130
## X51         -62.96802980668
## X52         -78.97039204924
## X53          -8.54225641510
## X54          28.66814057769
## X55         -14.23089446040
## X56          72.74811430234
## X57           4.48529136986
## X58          -2.39057970297
## X59          77.47038921163
## X60          -2.08781949092
## X61          -4.81420422516
## X62          -5.50703021308
## X63         -13.14311379365
## X64          11.49899429745
## X65          -0.00057320931
## X66          -0.00051863842
## X67          -0.00020465717
## X68          -0.00004878772
## X69          -0.00027621292
## X70          -0.00052362621
## X71         -33.01518029121
## X72          -9.04054325787
## X73         -12.54729474338
## X74         -17.29190283596
## X75         -12.33370428880
## X76           0.00016587262
## X77           0.00013818700
## X78           0.00011241482
## X79           0.00009267081
## X80           0.00016158623
```

```
## X81           0.00006911783
## X82           0.00004084222
## X83          -0.00004035699
## X84          -0.00011224087
## X85          -0.00005391842
## X86          -0.00003013422
## X87         -42.61359398964
## X88           7.27104681312
## X89           7.82257805523
## X90           6.11520300338
## X91           4.43647894288
## X92           5.05177460978
## X93           4.71051086688
## X94           4.83322033403
## X95           5.39841745342
## X96           5.33607589317
## X97           6.42067648638
## X98           7.13753041402
## X99           7.86283199351
## X100          3.77305303478
```

With 5-fold cross validation, the MSE of the best Lasso model where lambda ranges from 0 to 1 is 13.60371 as calculated in b, with lambda = 0.

4.4

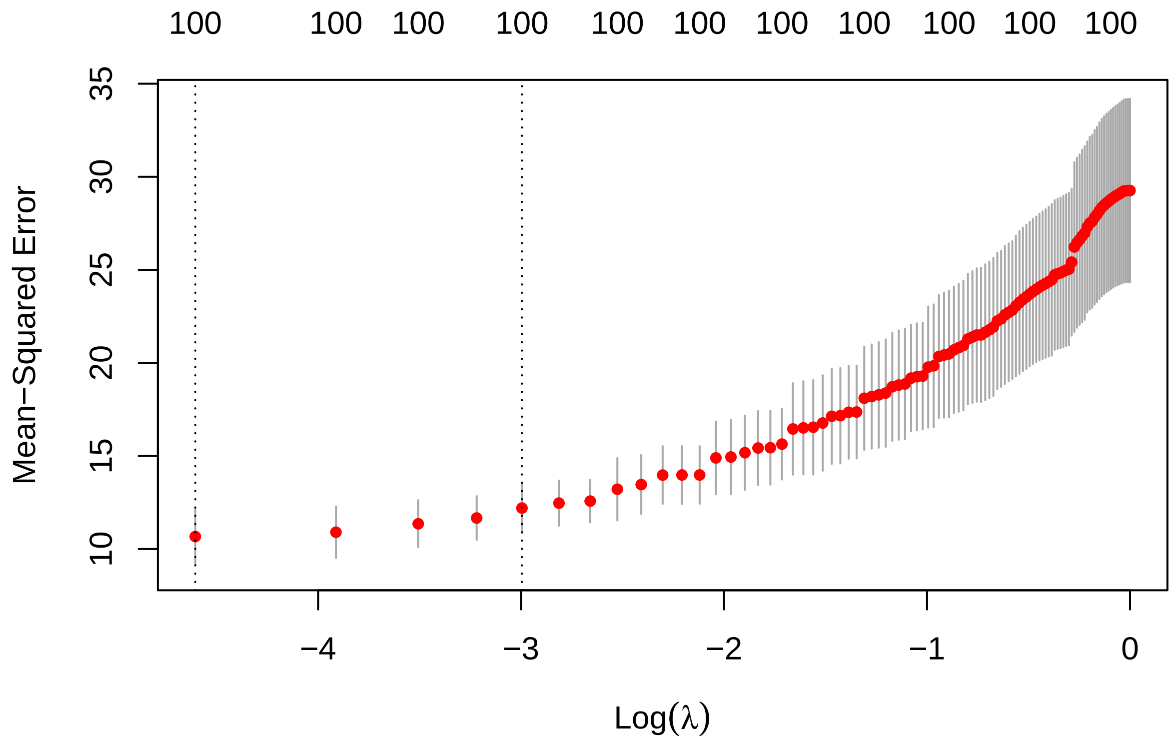
```r
# alpha = 0 (Ridge)
ridge <- glmnet(X, y, alpha = 0)
plot(ridge, xvar = "lambda", label = T)
```



```r
cv.ridge <- cv.glmnet(X, y, type = "mse", nfolds = 5, nlambda = 100, lambda = seq(0, 1, by = 0.01), alp
cv.ridge
```

```
##
## Call:  cv.glmnet(x = X, y = y, lambda = seq(0, 1, by = 0.01), type.measure = "mse",      nfolds = 5,
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min   0.01   100   10.67 1.541     100
## 1se   0.05    96   12.20 1.329     100
```

```r
plot(cv.ridge)
```




```r
coef(cv.ridge, s = "lambda.min")
```

```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  19.9432711
## X1           40.3608315
## X2           26.6506769
## X3           17.2794168
## X4           10.2135713
## X5            4.7961857
## X6            0.5793649
## X7           -2.7730942
## X8           -5.4241091
## X9           -7.6768084
## X10          -9.4637154
## X11         -10.8909147
```

```
## X12          -12.0220722
## X13          -12.9041983
## X14          -13.5424727
## X15          -13.9531493
## X16          -14.1055597
## X17          -14.0962039
## X18          -13.8721487
## X19          -13.3610740
## X20          -12.6470420
## X21          -11.6595023
## X22          -10.5373811
## X23           -9.3939194
## X24           -8.2630803
## X25           -7.2364570
## X26           -6.3936161
## X27           -5.5869877
## X28           -4.5390767
## X29           -3.0011738
## X30           -1.0367287
## X31            1.1677970
## X32            3.4789956
## X33            5.7862596
## X34            8.0857936
## X35           10.3171491
## X36           12.5573538
## X37           14.8722742
## X38           17.0526000
## X39           18.6293787
## X40           19.3512865
## X41           19.0980732
## X42           17.7689086
## X43           15.1352395
## X44           11.0101268
## X45            5.6711457
## X46           -0.1601710
## X47           -5.5244397
## X48           -9.4037597
## X49          -11.2009597
## X50          -11.2102509
## X51           -9.9961488
## X52           -7.8053611
## X53           -4.9399039
## X54           -1.8388927
## X55            0.9046726
## X56            2.8768261
## X57            3.8718937
## X58            4.0058566
## X59            3.3660410
## X60            2.7695399
## X61            2.0133849
## X62            1.3099630
## X63            0.5270859
## X64           -0.2216513
## X65           -0.9064106
```

```
## X66          -1.4395741
## X67          -1.7694084
## X68          -2.0741876
## X69          -2.5878357
## X70          -3.1351353
## X71          -3.4096255
## X72          -3.4150404
## X73          -3.5205399
## X74          -3.7221289
## X75          -3.6488169
## X76          -3.2035240
## X77          -2.9054436
## X78          -2.6006382
## X79          -2.2509559
## X80          -1.6749673
## X81          -1.0620827
## X82          -0.4276202
## X83           0.1515666
## X84           0.6108982
## X85           1.0224524
## X86           1.2426008
## X87           1.2919510
## X88           1.2033113
## X89           1.1480918
## X90           1.1696162
## X91           1.3674577
## X92           1.7525997
## X93           2.1516185
## X94           2.4860797
## X95           2.6981535
## X96           2.7548630
## X97           2.6344771
## X98           2.2392776
## X99           1.5030625
## X100          0.4773476
```

With 5-fold cross validation, the MSE of the best ridge model where lambda ranges from 0 to 1 is 13.60371 as calculated in b, with lambda = 0.

Q5

5.1

The ATE is 0. Because D and Y are independent. We can still run Y on D and we only include D, but the effect is not causal.

5.2

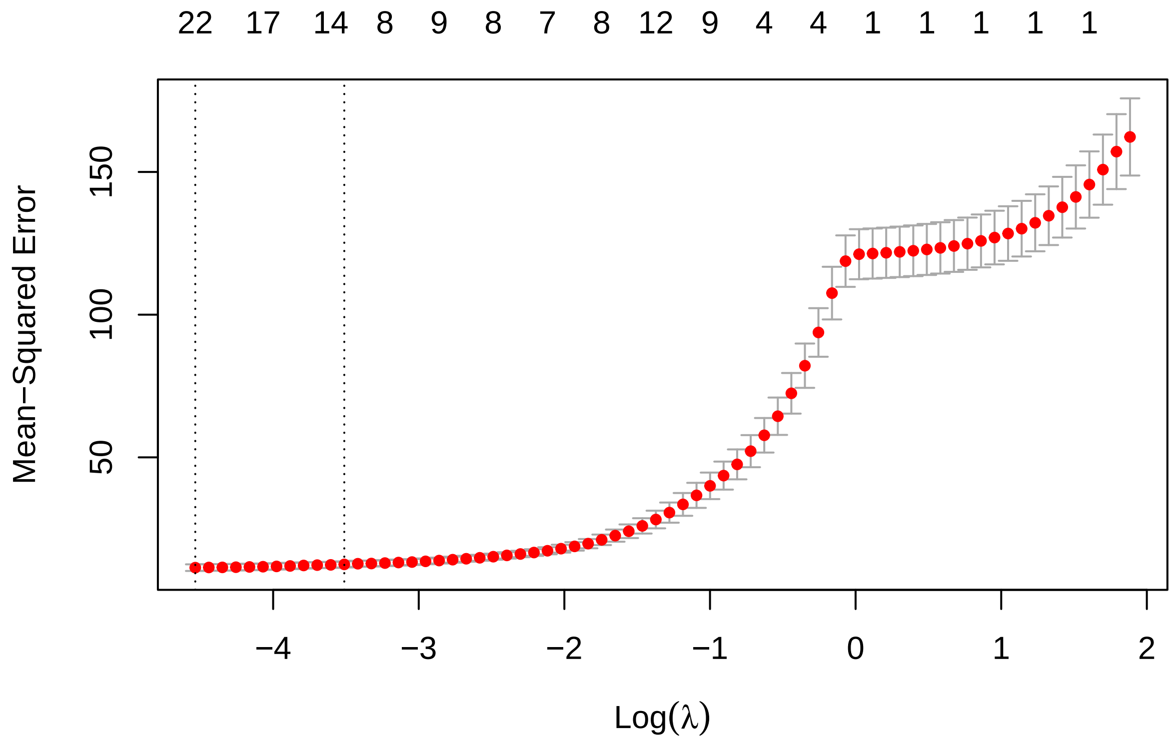
```r
lasso <- read.csv('lasso.csv')
cv.lasso1 <- cv.glmnet(X, y, type = "mse", alpha = 1)

cv.lasso1
```

```
##
## Call:  cv.glmnet(x = X, y = y, type.measure = "mse", alpha = 1)
```

```
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure    SE Nonzero
## min 0.01073    70   11.37 1.163      22
## 1se 0.02986    59   12.49 1.072      14
```

```r
plot(cv.lasso1, xvar = "lambda", label = T)
```

```
## Warning in plot.window(...): "xvar" is not a graphical parameter

## Warning in plot.window(...): "label" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "xvar" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "label" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter

## Warning in box(...): "xvar" is not a graphical parameter

## Warning in box(...): "label" is not a graphical parameter

## Warning in title(...): "xvar" is not a graphical parameter

## Warning in title(...): "label" is not a graphical parameter
```



```r
coef(cv.lasso1, s = "lambda.min")
```

```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  23.326350444
## X1           46.281285546
## X2             .
## X3             .
## X4             .
## X5             .
## X6             .
## X7             .
## X8             .
## X9             .
## X10            .
## X11            .
## X12            .
## X13            .
## X14         -46.391774302
## X15         -24.146945250
## X16         -13.805976718
## X17         -19.081269344
## X18         -12.459335761
## X19          -5.282139712
## X20          -5.498186167
## X21          -1.928623158
## X22          -0.905030785
## X23          -0.513157798
## X24          -2.978587681
## X25            .
## X26            .
## X27            .
## X28            .
## X29            .
## X30            .
## X31            .
## X32            .
## X33            .
## X34            .
## X35            .
## X36            .
## X37            .
## X38            .
## X39           0.002004740
## X40           2.924055010
## X41         139.783700985
## X42            .
## X43            .
## X44            .
## X45            .
## X46            .
## X47            .
## X48            .
## X49          -0.001367663
```

```
## X50          -0.176612221
## X51         -30.427759331
## X52         -33.872229527
## X53            .
## X54            .
## X55            .
## X56            .
## X57            .
## X58            .
## X59            .
## X60            .
## X61            .
## X62            .
## X63            .
## X64            .
## X65            .
## X66            .
## X67            .
## X68            .
## X69            .
## X70            .
## X71            .
## X72            .
## X73            .
## X74            .
## X75            .
## X76            .
## X77            .
## X78            .
## X79            .
## X80            .
## X81            .
## X82            .
## X83            .
## X84            .
## X85            .
## X86            .
## X87            .
## X88            .
## X89            .
## X90            .
## X91            .
## X92            .
## X93            .
## X94            .
## X95            .
## X96            .
## X97            .
## X98            0.681081919
## X99            3.400822636
## X100           0.912022256
```

There are 22 variables selected by Lasso, as shown above.

5.3

No, this is not a good estimate of the causal effect because the coefficients are only used for prediction. It is not specific to Lasso.