STAT435_HW1

Liyuan Tang

4/10/2020

Part 1

(A)

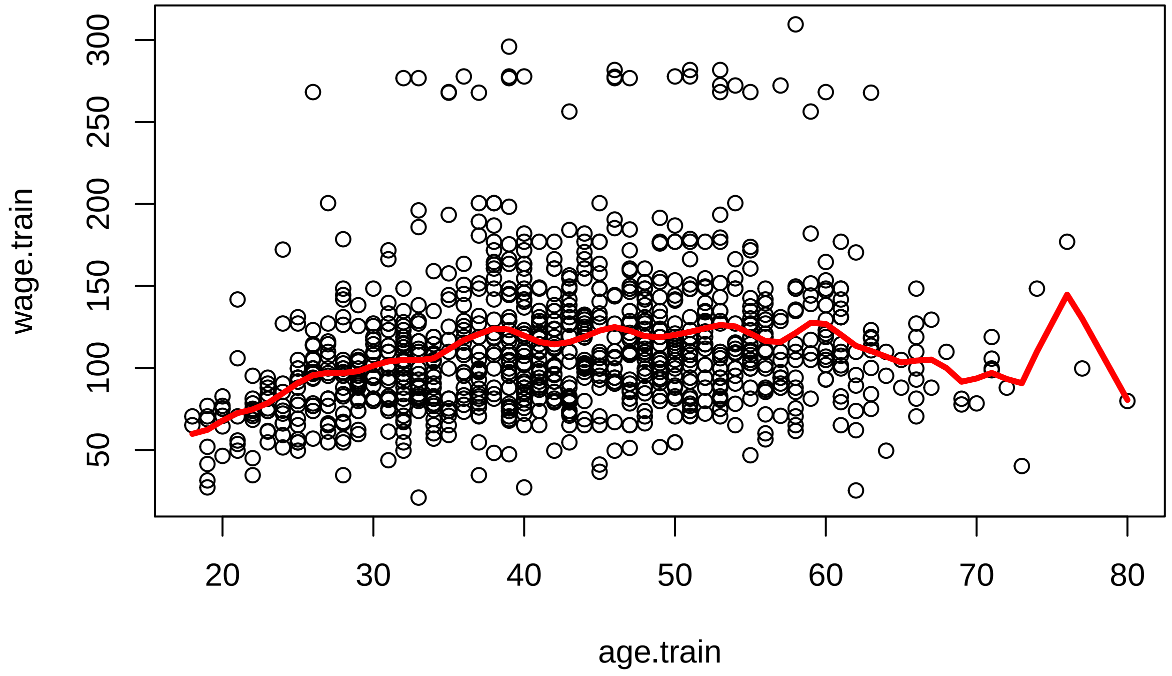
```r
source("home1-part1-data.R")
ksmooth.train <- function(x.train, y.train, kernel = c("box", "normal"),
bandwidth = 0.5, CV = False) {
  yhat.train <- list(rep(0, length(x.train)))
  if (kernel == "normal") {
    # find sigma
    sig = -0.25* bandwidth / qnorm(0.25, 0, 1)
    for (i in 1:length(x.train)) {
      x_curr = x.train[i]
      y_curr = y.train[i]
      y_abv = 0
      y_btm = 0
      for (j in 1:length(x.train)) {
        # if cv=T, i should not be used.
        if ( (CV & i!=j) | (!CV) ) {
          y_abv = y_abv + y.train[j] * dnorm(x_curr - x.train[j], 0, sig)
          y_btm = y_btm + dnorm(x_curr - x.train[j], 0, sig)
        }
      }
      yhat.train[i] = y_abv / y_btm
    }
  } else {
    #box kernel
    for (i in 1:length(x.train)) {
      total = c()
      for (j in 1:length(x.train)) {
        if ( ((!CV) | (CV & j!=i)) & abs(x.train[i] - x.train[j]) <= 0.5*bandwidth) {
          total = c(total, y.train[j])
        }
      }
      yhat.train[i] = sum(total) / length(total)
    }
  }
  yhat.train = unlist(yhat.train)
  df = data.frame(x.train, yhat.train)
}
```

(B)

```r
ksmooth.predict <- function(ksmooth.train.out, x.query) {
  x = ksmooth.train.out$x.train
  y = ksmooth.train.out$yhat.train
  x_min = min(x)
  x_max = max(x)
  y_min = y[match(x_min, x)]
  y_max = y[match(x_max, x)]
  y_predict = c()
  for (i in x.query) {
    if (i >= x_min & i <= x_max) {
      # linear interpolate
      y_predict = c(y_predict, approx(x, y, xout = i, method = "linear")$y)
    } else {
      #constant extrapolate
      if (i < x_min) {
        y_predict = c(y_predict, y_min)
      } else {
        y_predict = c(y_predict, y_max)
      }
    }
  }
  return(data.frame(x.query, y_predict))
}
```

(C)

```r
wage.train = Wage.train$wage
age.train = Wage.train$age
plot(age.train, wage.train)
ksmooth.train.out = ksmooth.train(age.train, wage.train, "normal", 3, FALSE)
result_df = ksmooth.train.out[order(ksmooth.train.out[,1]),]
lines(result_df$x.train, result_df$yhat.train,lwd = 3, col = 'red')
```



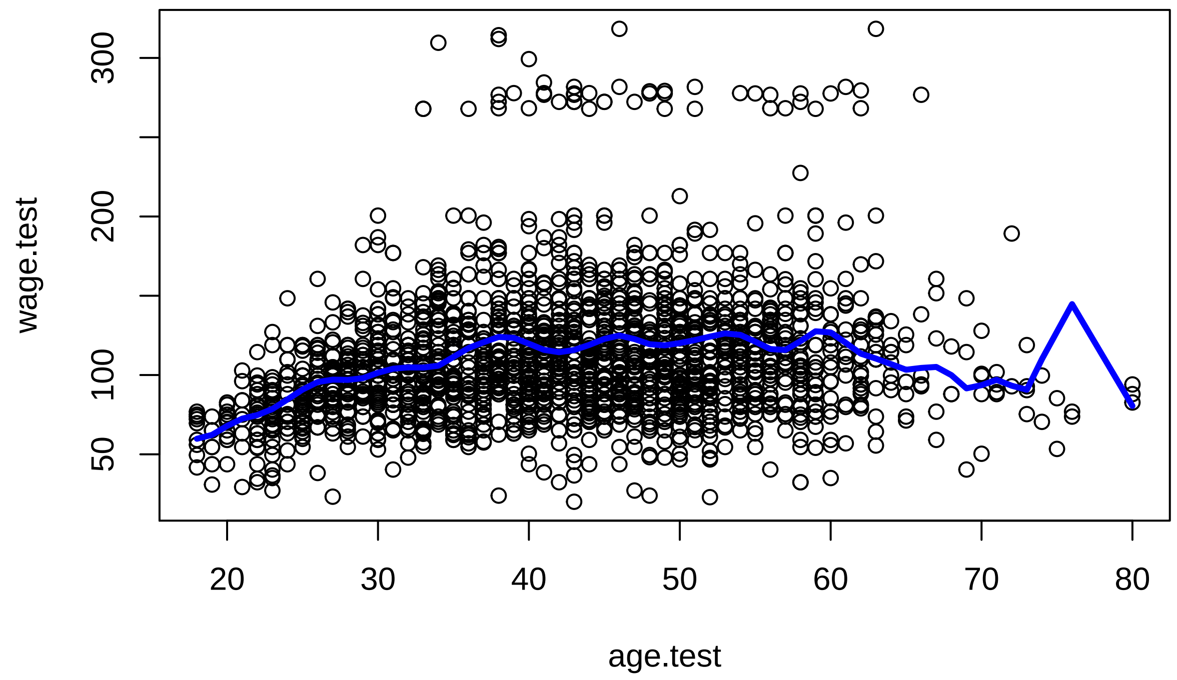
```r
# Calculate residual sum of squares
origin_df = data.frame(age.train, wage.train)
origin_df = origin_df[order(origin_df[,1]),]

rss = sum((origin_df$wage.train - result_df$yhat.train)^2)
cat("the residual sum of squares is", rss)
```

```
## the residual sum of squares is 1625121
```

```r
# 1625120.9467
```

(D)

```r
wage.test = Wage.test$wage
age.test = Wage.test$age
wage.predict = ksmooth.predict(result_df,age.test)
plot(age.test, wage.test)
predict_df = data.frame(age.test, wage.predict$y_predict)
predict_df = predict_df[order(predict_df[,1]),]
lines(predict_df, lwd = 3, col = "blue")
```



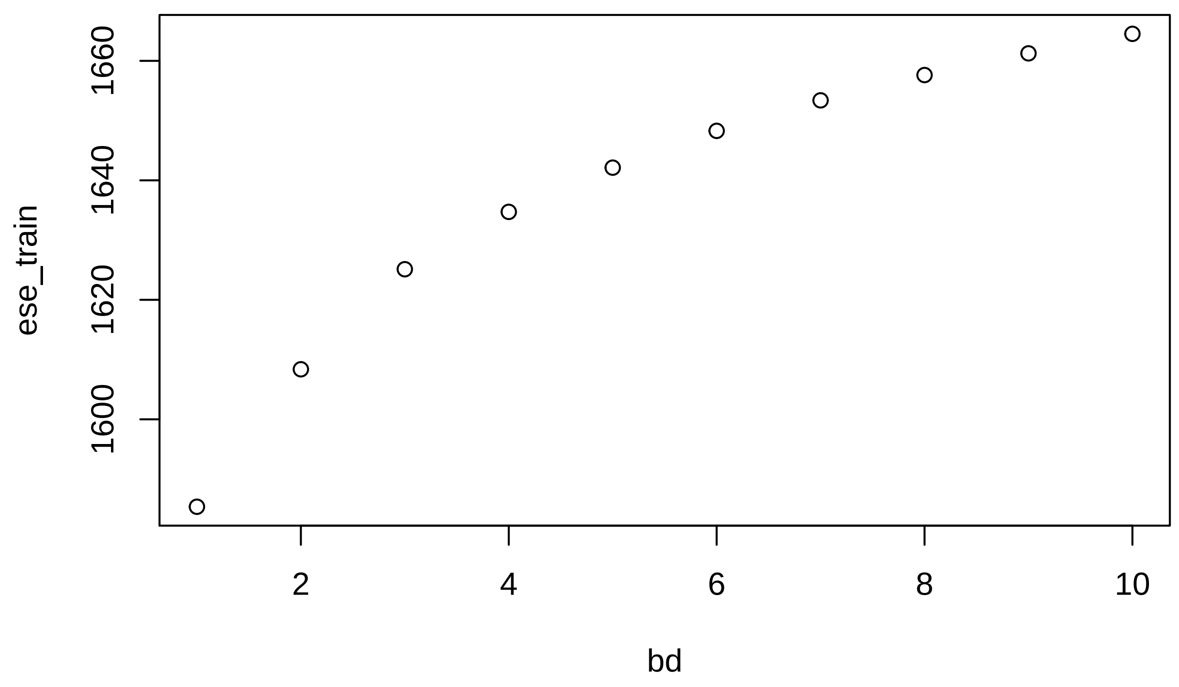
```r
# Calculate residual sum of squares
origin_df = data.frame(age.test, wage.test)
origin_df = origin_df[order(origin_df[,1]),]

rss = sum((origin_df$wage.test - predict_df$wage.predict)^2)
cat("the residual sum of squares is", rss)
```

```
## the residual sum of squares is 3168000
```

```r
# 3168000.16699
```

(E)

```r
bd = seq(1, 10, 1)
ese_train = c()
for (i in bd) {
  train_df = ksmooth.train(age.train, wage.train, "normal", i, FALSE)
  ese_train = c(ese_train, sum((train_df$yhat.train-wage.train)^2) / length(wage.train))
}
plot(bd, ese_train)
```

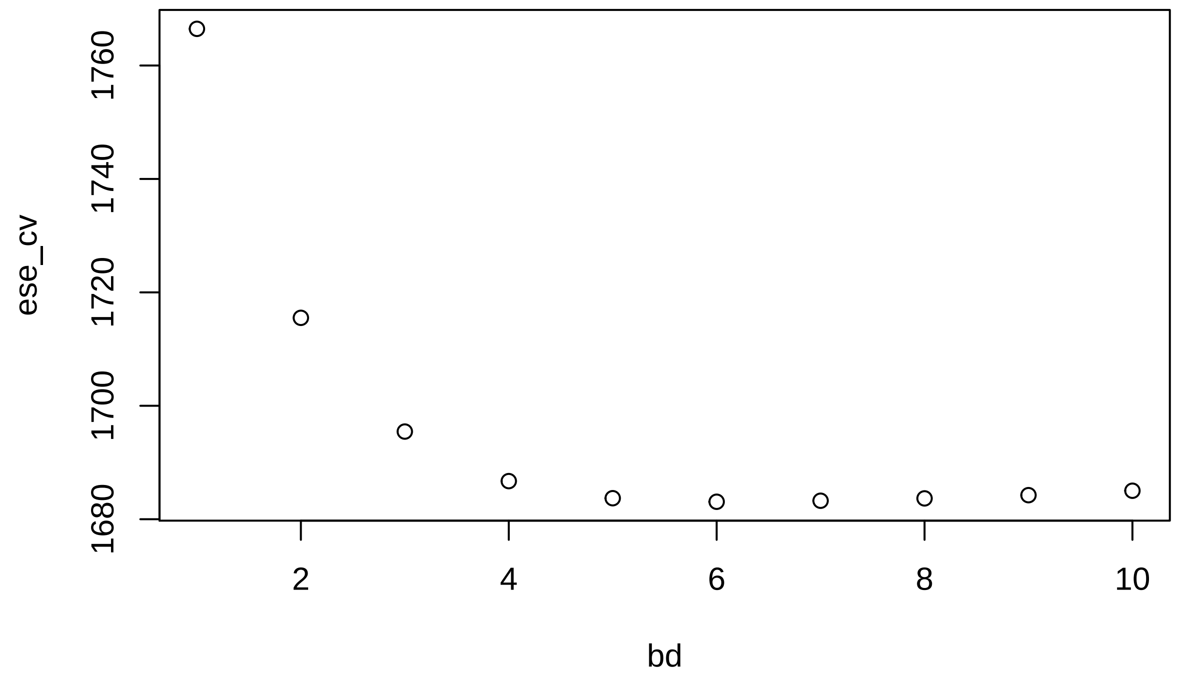


```r
print(ese_train)
```

```
##  [1] 1585.364 1608.370 1625.121 1634.722 1642.120 1648.282 1653.387 1657.624
##  [9] 1661.252 1664.519
```

(F)

```r
ese_cv = c()
for (i in bd) {
  train_df = ksmooth.train(age.train, wage.train, "normal", i, TRUE)
  ese_cv = c(ese_cv, sum((train_df$yhat.train-wage.train)^2) / length(wage.train))
}
plot(bd, ese_cv)
```



```r
print(ese_cv)
```

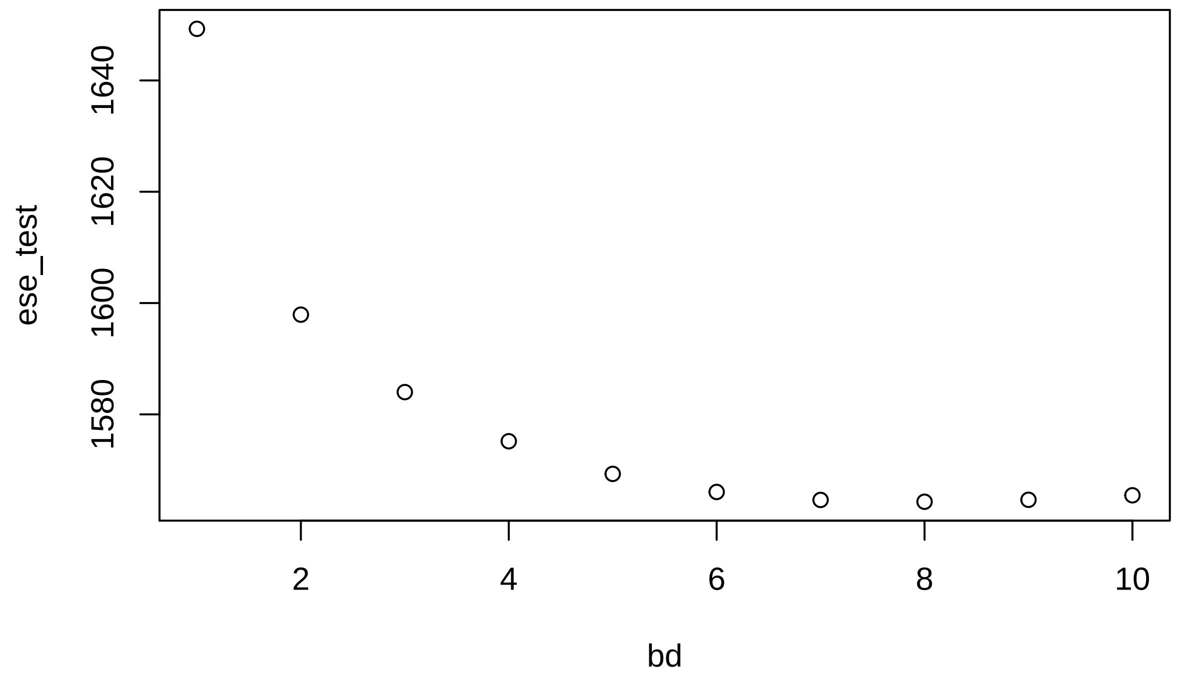
```
##  [1] 1766.466 1715.508 1695.453 1686.715 1683.705 1683.079 1683.256 1683.671
##  [9] 1684.239 1685.021
```

```r
hopt = match(min(ese_cv), ese_cv)
cat(hopt, "is the bandwidth I would choose")
```

```
## 6 is the bandwidth I would choose
```

(G)

```r
ese_test = c()
for (i in bd) {
  train_df = ksmooth.train(age.train, wage.train, "normal", i, FALSE)
  wage.predict = ksmooth.predict(train_df,age.test)
  ese_test = c(ese_test, sum((wage.predict$y_predict-wage.test)^2) / length(wage.test))
}
plot(bd, ese_test)
```

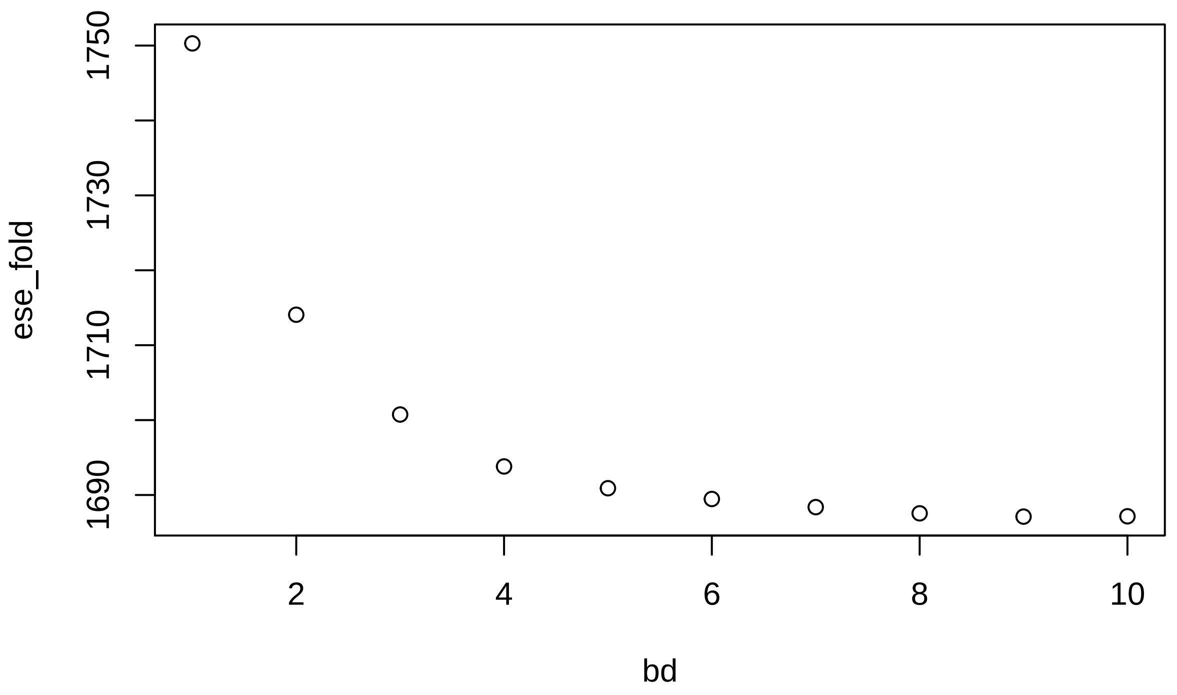


```
print(ese_test)
```

```
##  [1] 1649.268 1597.913 1584.000 1575.168 1569.304 1566.052 1564.621 1564.297
##  [9] 1564.647 1565.453
```

(H)

```
ese_fold = c()
k = 5
for (i in bd) {
  ese_kfold = c()
  for (j in 1:k) {
    age.test = Wage.train$age[which(fold == j)]
    age.train = Wage.train$age[-which(fold == j)]
    wage.test = Wage.train$wage[which(fold == j)]
    wage.train = Wage.train$wage[-which(fold == j)]
    smooth_df = ksmooth.train(age.train, wage.train, "normal", i, FALSE)
    predict_wage = ksmooth.predict(smooth_df, age.test)
    ese_kfold = c(ese_kfold, sum((predict_wage$y_predict - wage.test)^2)/length(wage.test))
  }
  ese_fold[i] = sum(ese_kfold) / k
}
plot(bd, ese_fold)
```



```r
print(ese_fold)
```

```
##  [1] 1750.292 1714.070 1700.746 1693.811 1690.903 1689.468 1688.382 1687.550
##  [9] 1687.115 1687.157
```

Part 2

(A) Notice that $\hat{\mathbf{f}} = \mathbf{W}\mathbf{y} = \mathbf{W}(\mathbf{f} + \epsilon)$

D $= E(\frac{1}{n}||\mathbf{f} - \hat{\mathbf{f}}||^2)$

$= \frac{1}{n}E((\mathbf{f} - \hat{\mathbf{f}})^{\mathbf{T}}(\mathbf{f} - \hat{\mathbf{f}}))$

$= \frac{1}{n}E(\mathbf{f}^{\mathbf{T}}\mathbf{f} - \mathbf{2f}^{\mathbf{T}}\hat{\mathbf{f}} + \hat{\mathbf{f}}^{\mathbf{T}}\hat{\mathbf{f}})$

$= \frac{1}{n}E(\mathbf{f}^{\mathbf{T}}\mathbf{f} - \mathbf{2f}^{\mathbf{T}}\mathbf{W}(\mathbf{f} + \epsilon) + (\mathbf{f}^{\mathbf{T}} + \epsilon^{\mathbf{T}})\mathbf{W}^{\mathbf{T}}\mathbf{W}(\mathbf{f} + \epsilon))$

$= \frac{1}{n}E(\mathbf{f}^{\mathbf{T}}\mathbf{f} - \mathbf{2f}^{\mathbf{T}}\mathbf{W}\mathbf{f} + \mathbf{f}^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\mathbf{f}) + \frac{1}{n}E(\epsilon^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon) - \frac{2}{n}E(\mathbf{f}^{\mathbf{T}}\mathbf{W}\epsilon) + \frac{2}{n}E(\mathbf{f}^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon)$

Consider $E(\mathbf{f}^{\mathbf{T}}\mathbf{W}\epsilon)$. Define $V = \mathbf{f}^{\mathbf{T}}\mathbf{W}$, so $E(\mathbf{f}^{\mathbf{T}}\mathbf{W}\epsilon) = E(V\epsilon) = E(\sum_i V_i\epsilon_i) = 0$ due to the linearity. And similar for the $E(\mathbf{f}^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon)$.

So D $= \frac{1}{n}(E(\mathbf{f}^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\mathbf{f} - \mathbf{2f}^{\mathbf{T}}\mathbf{W}\mathbf{f} + \mathbf{f}^{\mathbf{T}}\mathbf{f}) + E(\epsilon^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon))$

$= \frac{1}{n}(||\mathbf{W}\mathbf{f} - \mathbf{f}||^2 + E(\epsilon^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon))$

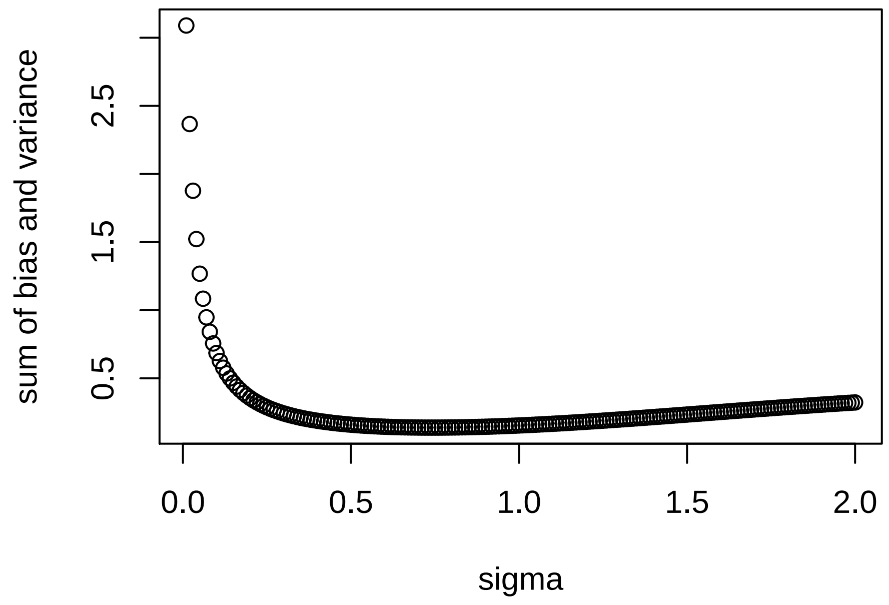
$= \frac{1}{n}(||(\mathbf{W} - \mathbf{I})\mathbf{f}||^2 + E(\epsilon^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon))$

Set the vector $Y = W\epsilon$. Then $E(\epsilon^{\mathbf{T}}\mathbf{W}^{\mathbf{T}}\mathbf{W}\epsilon) = E(Y^TY) = E(\sum_{i=1}^n Y_i^2) = \sum_{i=1}^n E(Y_i^2) = \sum_{i=1}^n Var(Y_i) = \sum_{i=1}^n \sum_{j=1}^n \mathbf{W}_{ij}^2\sigma^2 = \sigma^2\mathbf{trace}(\mathbf{W}^{\mathbf{T}}\mathbf{W})$

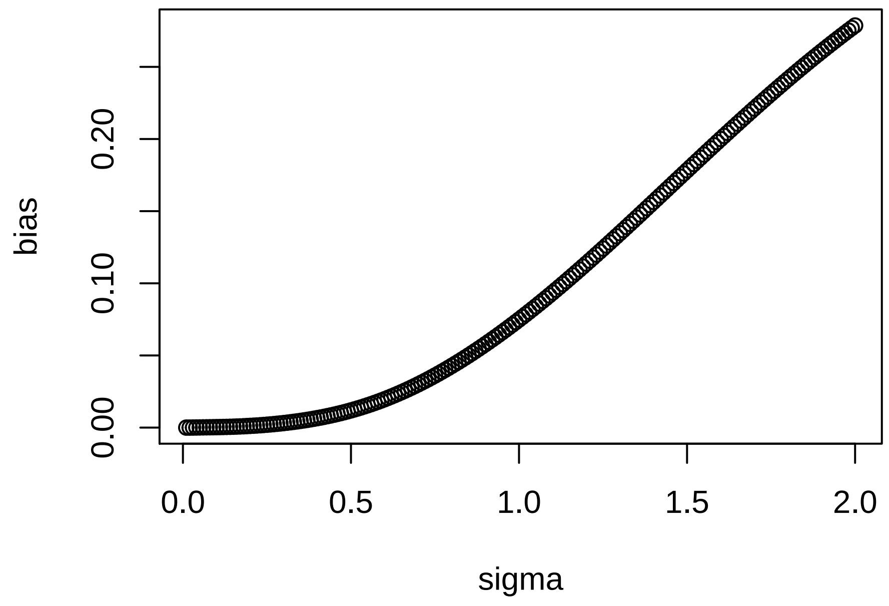
Thus, D = $\frac{1}{n}(||(\mathbf{W} - \mathbf{I})\mathbf{f}||^2 + \sigma^2\mathbf{trace}(\mathbf{W}^{\mathbf{T}}\mathbf{W}))$

(B)

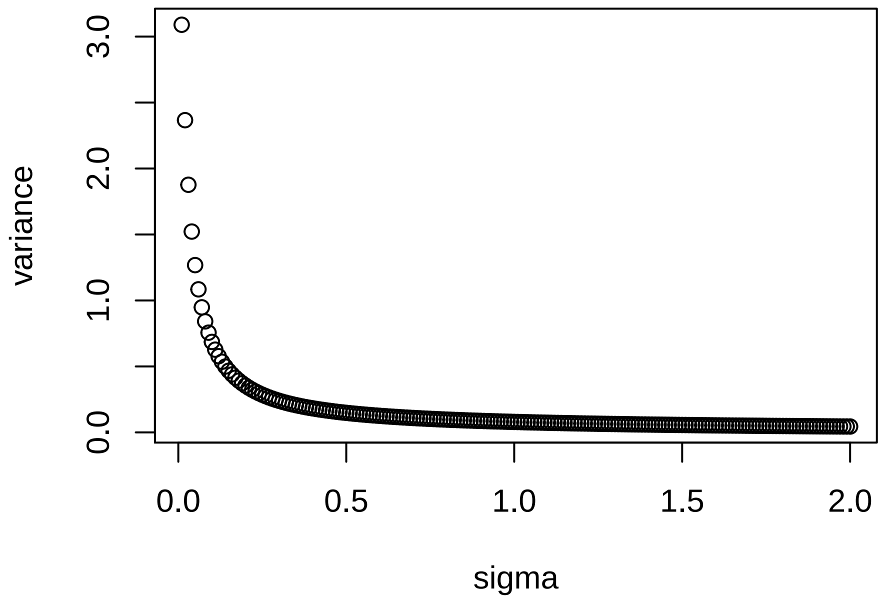
```
source("home1-part2-data.R")
sig_vec = seq(from = 0.01, to = 2, by = 0.01)
n = length(x.train)
bias.vec = numeric(n)
var.vec = numeric(n)


for (k in 1:length(sig_vec)) {
  sig = sig_vec[k]
  # W matrix
  W.matrix = matrix(data = NA, nrow = n, ncol = n)
  for (i in 1:n) {
    bt = sum(dnorm(x.train[i] - x.train, 0 , sig))
    for (j in 1:n) {
      W.matrix[i,j] = dnorm(x.train[i] - x.train[j], 0, sig) / bt
    }
  }

  # bias
  bias.mat = (W.matrix - diag(1,n,n)) %*% f
  bias.vec[k] = norm(x = bias.mat, type = "2")^2 / n

  # variance
  var.vec[k] = sum(diag(t(W.matrix) %*% W.matrix)) * noise.var / n
}
plot(sig_vec, bias.vec + var.vec, xlab = "sigma", ylab = "sum of bias and variance")
```



```r
plot(sig_vec, bias.vec, xlab = "sigma", ylab = "bias")
```



```r
plot(sig_vec, var.vec, xlab = "sigma", ylab = "variance")
```

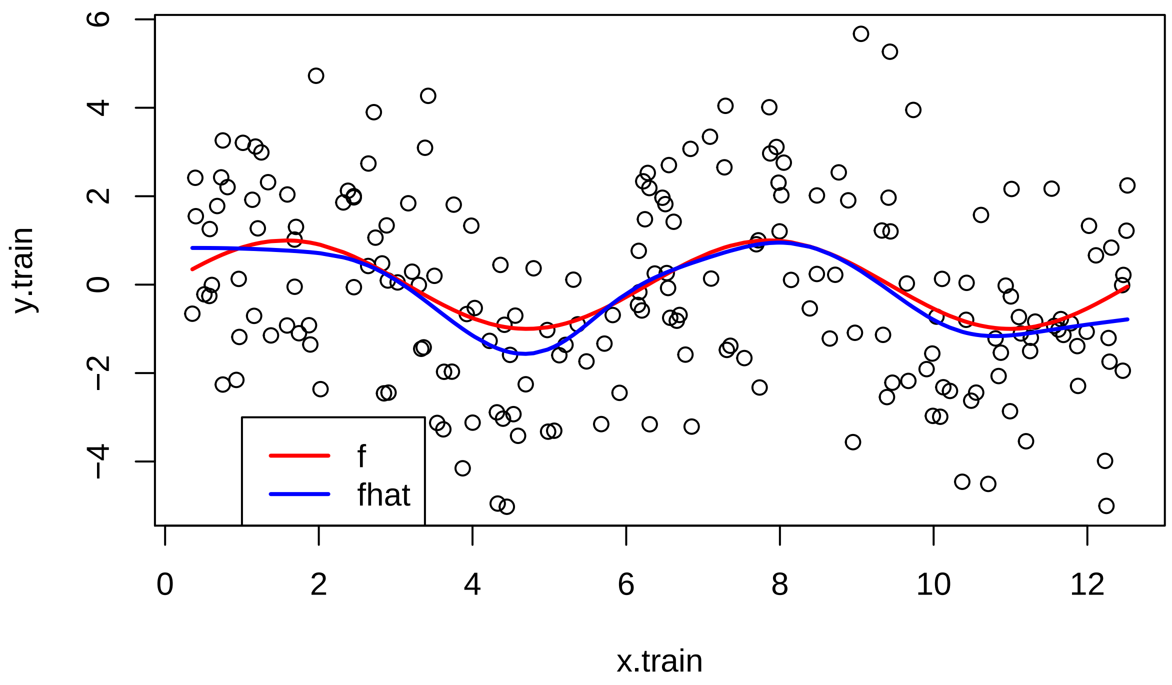


```r
D = bias.vec + var.vec
sig_opt = sig_vec[which(D == min(D))]
cat("The optimal choice of sigma is", sig_opt)
```

```
## The optimal choice of sigma is 0.74
```

(C)

```r
W.matrix = matrix(data = NA, nrow = n, ncol = n)
for (i in 1:n) {
  bt = sum(dnorm(x.train[i] - x.train, 0 , sig_opt))
  for (j in 1:n) {
    W.matrix[i,j] = dnorm(x.train[i] - x.train[j], 0, sig_opt) / bt
  }
}
fhat = W.matrix %*% y.train
plot(x.train, y.train)
lines(x.train, f,lwd = 2, col = "red")
lines(x.train, fhat, lwd = 2, col = "blue")
legend(1, -3, legend=c("f", "fhat"),
       col=c("red", "blue"), lty=c(1,1), lwd = 2)
```


y.train
x.train
f
fhat
6
4
2
0
−2
−4
0
2
4
6
8
10
12