a1

Mushi Wang

30/05/2020

q1 (a) Let $\beta = (\beta_1, \beta_2, \ldots, \beta_n)^T$, $X = (x_1, x_2, \ldots, x_n)^T$ and Σ be variance-covariance matrix.

$$\begin{aligned}
E(\tilde{\beta}_{WLS}) &= E((X^TWX)^{-1}X^TWY) \\
&= (X^TWX)^{-1}X^TWE(Y) \\
&= (X^TWX)^{-1}X^TWE(X\beta + \epsilon) \\
&= (X^TWX)^{-1}X^TWXE(\beta) + (X^TWX)^{-1}XWE(\epsilon) \\
&= (X^TWX)^{-1}X^TWX\beta + 0 \qquad (E(\epsilon) = 0) \\
&= \beta
\end{aligned}$$

(b) Let $W = diag(\frac{1}{g(x_1)}, \frac{1}{g(x_n)}, \ldots, \frac{1}{g(x_n)})$. So $W^T = W$.
Since $\Sigma = \sigma^2 diag(g(x_1), g(x_2), \ldots, g(x_n))$, so $\Sigma \times W = \sigma^2$

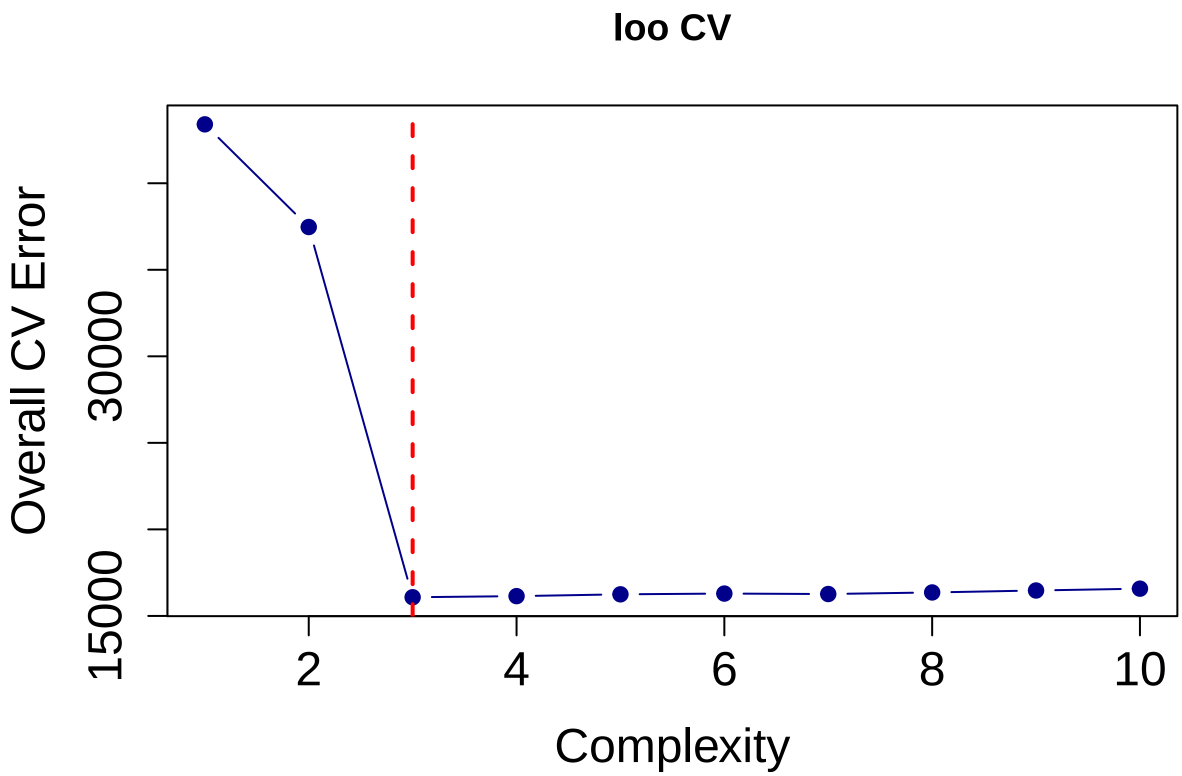
$$\begin{aligned}
Var(\tilde{\beta}_{WLS}) &= Var((X^TWX)^{-1}XWY) \\
&= (X^TWX)^{-1}X^TW \times Var(Y) \times ((X^TWX)^{-1}X^TW)^T \\
&= (X^TWX)^{-1}X^TW \times Var(X\beta + \epsilon) \times ((X^TWX)^{-1}X^TW)^T \\
&= (X^TWX)^{-1}X^TW \times Var(\epsilon) \times W^TX(X^TW^TX)^{-1} \\
&= (X^TWX)^{-1}X^TW \times \Sigma \times W^TX(X^TW^TX)^{-1} \\
&= \sigma^2(X^TWX)^{-1}X^TWX(X^TW^TX)^{-1} \\
&= \sigma^2(X^TWX)^{-1}
\end{aligned}$$

q2

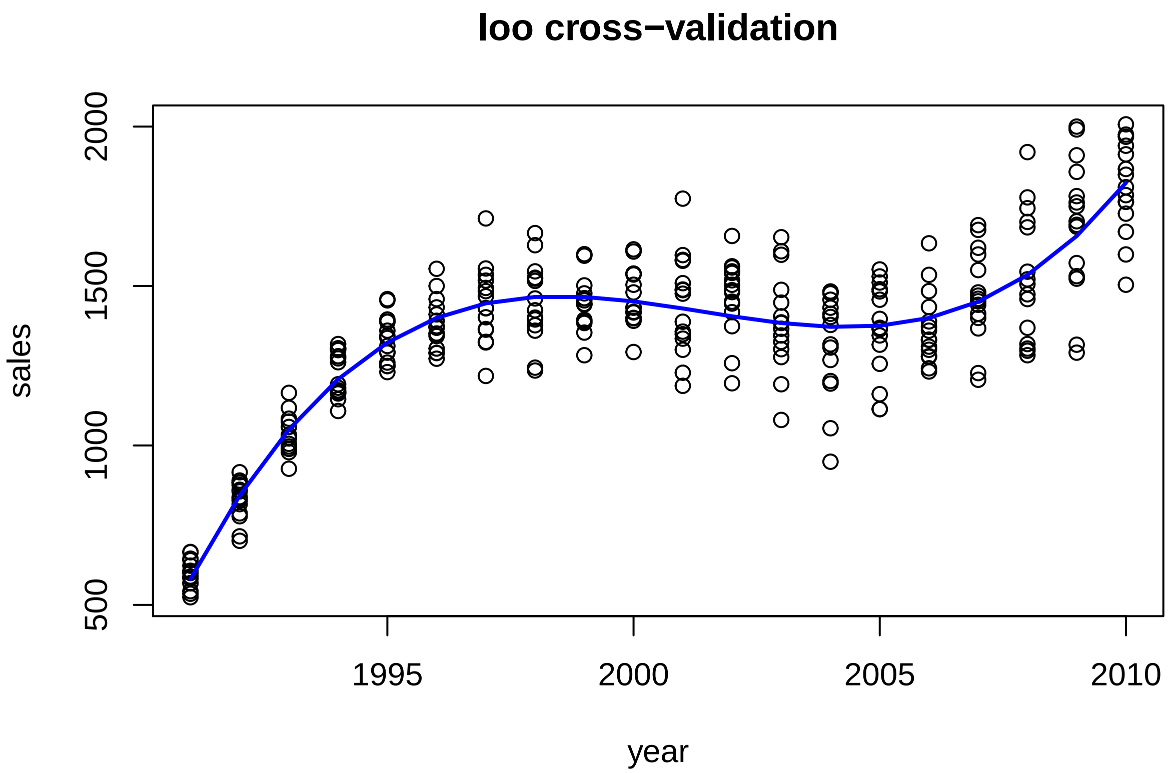
```r
sales = read.table("JaxSales.txt", header = TRUE)

# A function to generate the indices of the k-fold sets
kfold <- function(N, k=N, indices=NULL){
  # get the parameters right:
  if (is.null(indices)) {
    # Randomize if the index order is not supplied
    indices <- sample(1:N, N, replace=FALSE)
  } else {
    # else if supplied, force N to match its length
    N <- length(indices)
  }
  # Check that the k value makes sense.
  if (k > N) stop("k must not exceed N")
  #

  # How big is each group?
  gsize <- rep(round(N/k), k)

  # For how many groups do we need odjust the size?
  extra <- N - sum(gsize)

  # Do we have too few in some groups?
  if (extra > 0) {
    for (i in 1:extra) {
      gsize[i] <- gsize[i] +1
    }
  }
  # Or do we have too many in some groups?
  if (extra < 0) {
    for (i in 1:abs(extra)) {
      gsize[i] <- gsize[i] - 1
    }
  }

  running_total <- c(0,cumsum(gsize))

  # Return the list of k groups of indices
  lapply(1:k,
         FUN=function(i) {
           indices[seq(from = 1 + running_total[i],
                       to = running_total[i+1],
                       by = 1)
                  ]
         }
  )
}


# A function to form the k samples
getKfoldSamples <- function (x, y, k, indices=NULL){
  groups <- kfold(length(x), k, indices)
```

```r
  #training sets
  Ssamples <- lapply(groups,
                     FUN=function(group) {
                       list(x=x[-group], y=y[-group])
                     })
  #test set
  Tsamples <- lapply(groups,
                     FUN=function(group) {
                       list(x=x[group], y=y[group])
                     })
  list(Ssamples = Ssamples, Tsamples = Tsamples)
}
```

```r
# For leave one out cross-validation
samples_loocv <-  getKfoldSamples(sales$Year, sales$Sales, k=length(sales$Sales))

# the degrees of freedom associated with each
complexity <- c(1:10) # These are the degrees of polynomials to be fitted

# Performing the Cross-Validation
Ssamples <- samples_loocv$Ssamples # change this accorcing to the number of folds
Tsamples <- samples_loocv$Tsamples # change this accorcing to the number of folds
CV.To.Plot = data.frame(Complexity=NA , MSE=NA)
for(i in 1:length(complexity)){
  MSE = c()
  for(j in 1:length(Ssamples)){
    x.temp = Ssamples[[j]]$x
    y.temp = Ssamples[[j]]$y
    model = lm(y.temp~poly(x.temp, complexity[i]))
    pred = predict(model, newdata=data.frame(x.temp=Tsamples[[j]]$x))
    MSE[j] = mean((Tsamples[[j]]$y-pred)^2)
  }
  CV.To.Plot[i,] = c(complexity[i], mean(MSE))
}


Title.Graph = "loo CV" # change this accorcing to the number of folds
plot(CV.To.Plot, pch=19, col="darkblue", type="b",
     cex.axis = 1.5, cex.lab=1.5, ylab="Overall CV Error")
indx = which.min(CV.To.Plot$MSE)
abline(v=indx, lty=2, lwd=2, col='red')
title(main=Title.Graph)
```



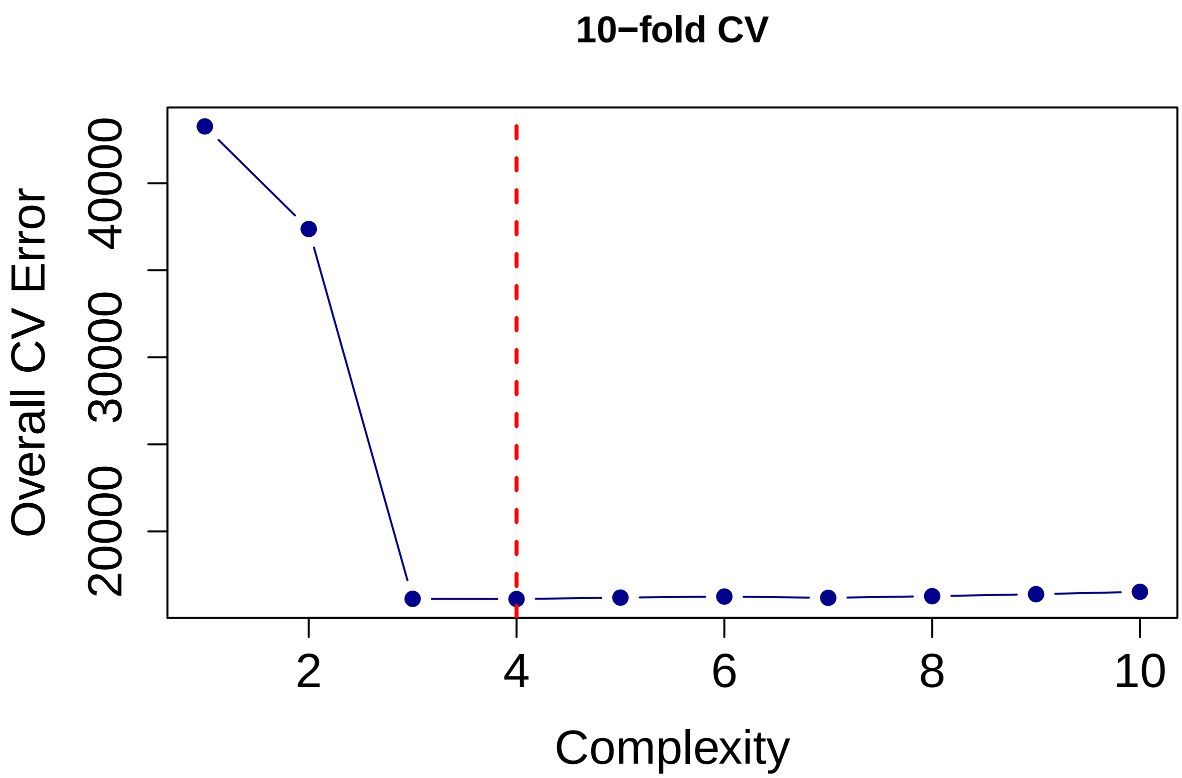

```r
plot(sales$Year, sales$Sales, xlab = "year", ylab = "sales", main = "loo cross-validation")
lines(sales$Year, predict(lm(sales$Sales~poly(sales$Year,3))), type="l", col="blue", lwd=2)
```



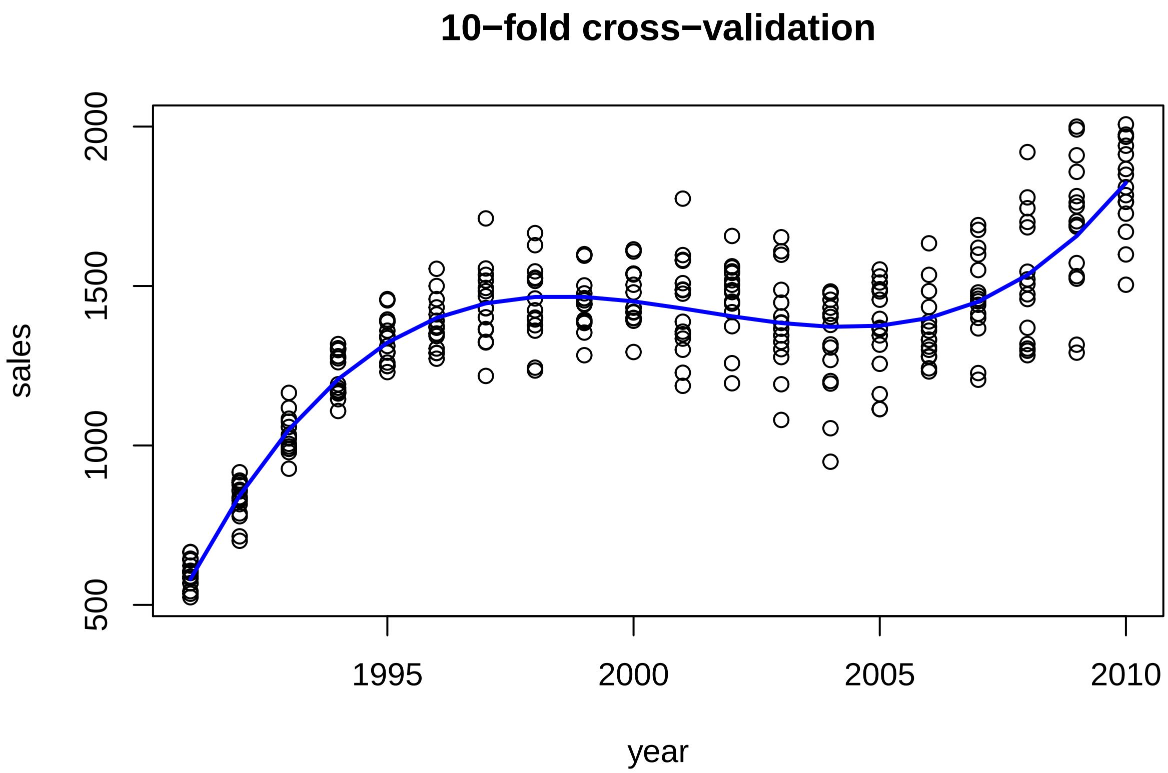
(b)

```r
# For leave one out cross-validation
samples_10fold <- getKfoldSamples(sales$Year, sales$Sales, k=10)

# the degrees of freedom associated with each
complexity <- c(1:10) # These are the degrees of polynomials to be fitted

# Performing the Cross-Validation
Ssamples <- samples_10fold$Ssamples # change this accorcing to the number of folds
Tsamples <- samples_10fold$Tsamples # change this accorcing to the number of folds
CV.To.Plot = data.frame(Complexity=NA , MSE=NA)
for(i in 1:length(complexity)){
  MSE = c()
  for(j in 1:length(Ssamples)){
    x.temp = Ssamples[[j]]$x
    y.temp = Ssamples[[j]]$y
    model = lm(y.temp~poly(x.temp, complexity[i]))
    pred = predict(model, newdata=data.frame(x.temp=Tsamples[[j]]$x))
    MSE[j] = mean((Tsamples[[j]]$y-pred)^2)
  }
  CV.To.Plot[i,] = c(complexity[i], mean(MSE))
}


Title.Graph = "10-fold CV" # change this accorcing to the number of folds
plot(CV.To.Plot, pch=19, col="darkblue", type="b",
```

```r
     cex.axis = 1.5, cex.lab=1.5, ylab="Overall CV Error")
indx = which.min(CV.To.Plot$MSE)
abline(v=indx, lty=2, lwd=2, col='red')
title(main=Title.Graph)
```



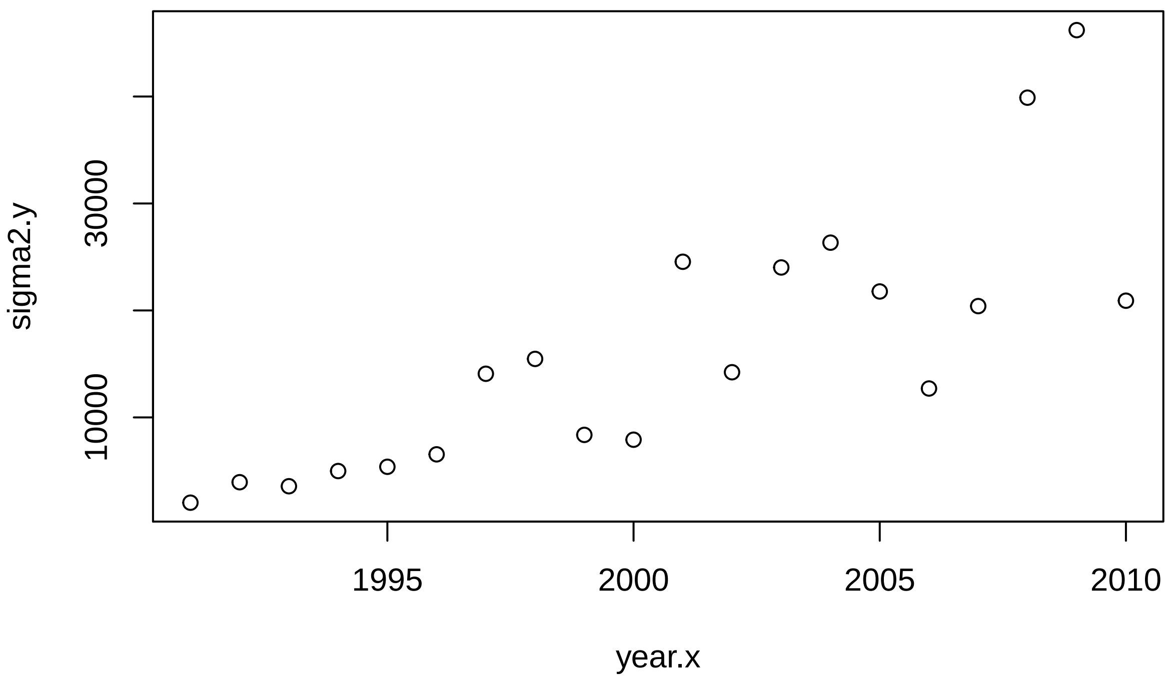

```r
plot(sales$Year, sales$Sales, xlab = "year", ylab = "sales", main = "10-fold cross-validation")
lines(sales$Year, predict(lm(sales$Sales~poly(sales$Year, 3))), type="l", col="blue", lwd=2)
```



(c) The two models above result in same model where complexity is 3. I prefer $k = 10$. Even though they result in the same model and LOO cross-validation is approximately unbiased. However, LOO cross-validation causes high variance.

q3 (a)

```
year.x = seq(1991, 2010)
sigma2.y = vector(length = 20)
for (i in year.x) {
  sigma2.y[i + 1 - 1991] = var(sales$Sales[which(sales$Year == i)])
}
plot(year.x, sigma2.y)
```



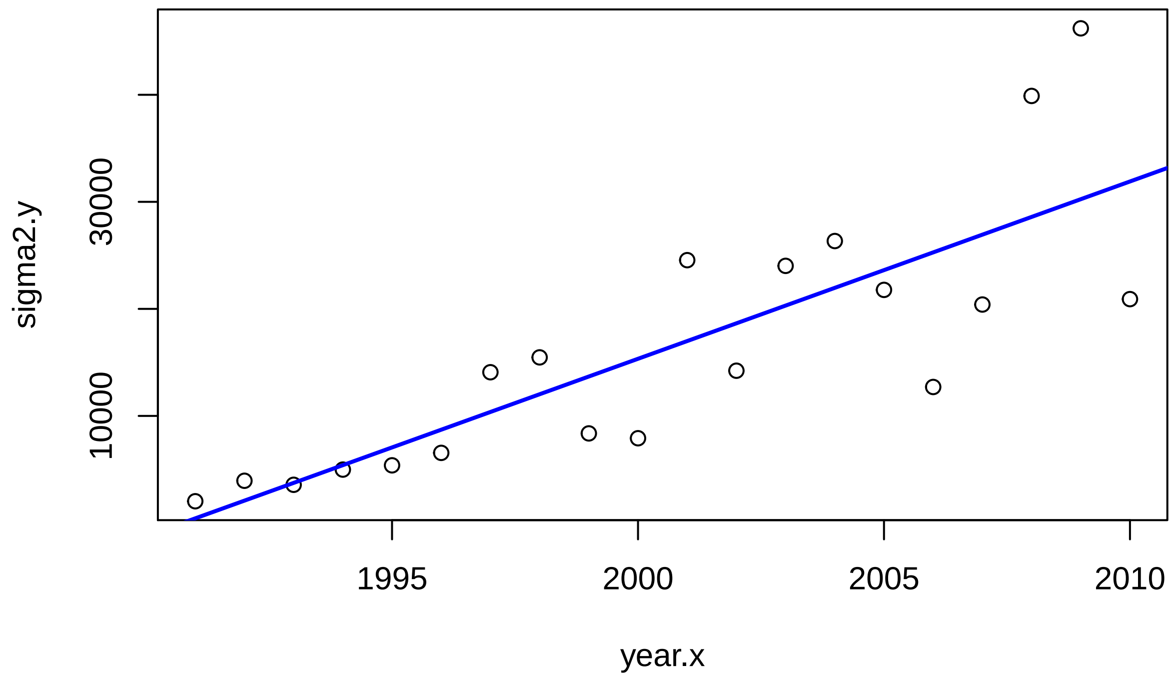
(b)

```
varModel = lm(sigma2.y ~ year.x)
varModel$coefficients
```

```
##  (Intercept)       year.x
## -3297581.598     1656.459
```

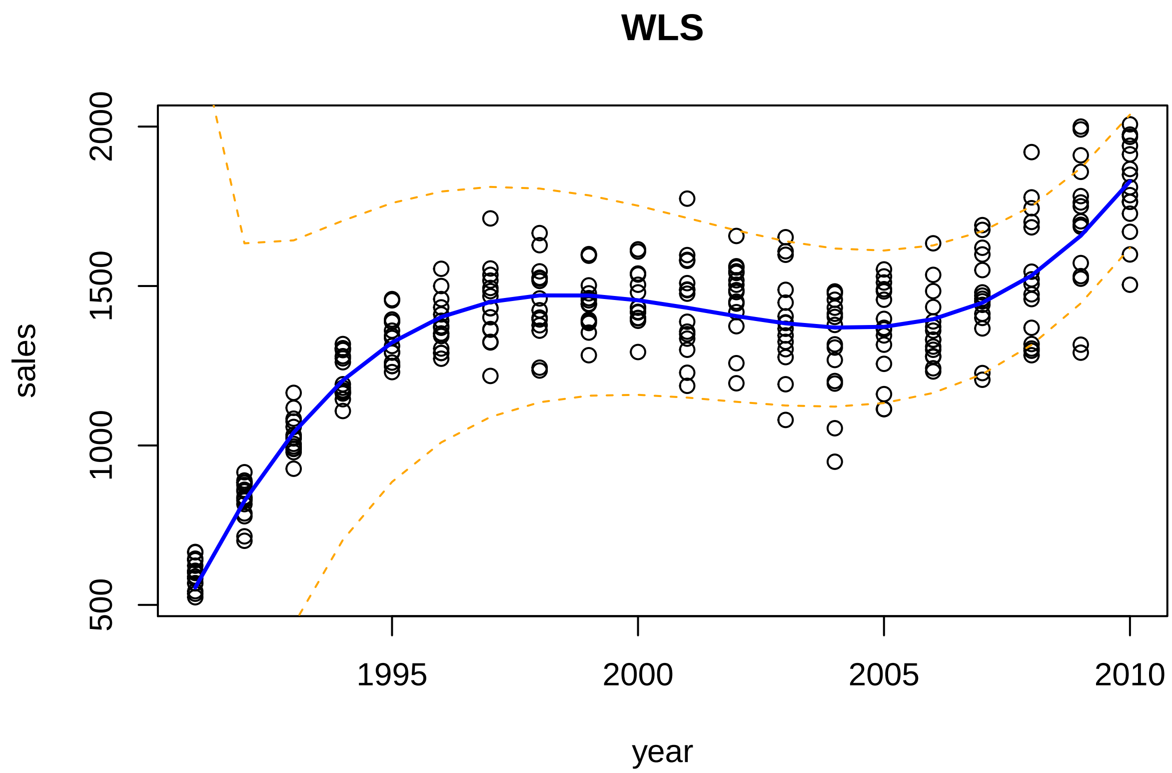
$\hat{\alpha}_0 = -3297581.598, \hat{\alpha}_1 = 1656.459$

```
plot(year.x, sigma2.y)
abline(lm(sigma2.y ~ year.x), col="blue", lwd = 2)
```



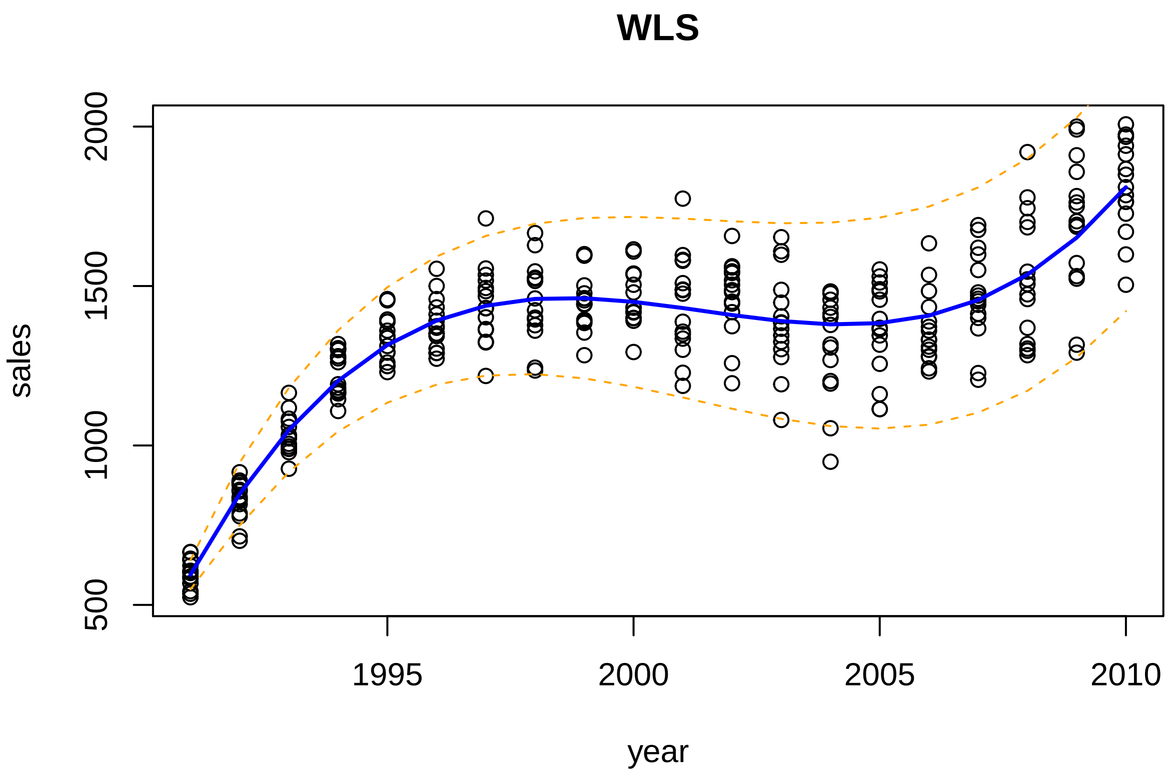
(c)(1)

```
w1 = vector(length = length(sales$Year))
w1 = varModel$coefficients[1] + varModel$coefficients[2] * sales$Year
wls1 = lm(sales$Sales~poly(sales$Year, 3), weight = w1)


plot(sales$Year, sales$Sales, xlab = "year", ylab = "sales", main = "WLS")
lines(sales$Year, predict(wls1), type="l", col="blue", lwd=2)

pred_interval1 <- predict(wls1, newdata = data.frame(year=sales$Year),
                          interval="prediction", level = 0.95, weight = w1)
lines(sales$Year, pred_interval1[,2], col="orange", lty=2)
lines(sales$Year, pred_interval1[,3], col="orange", lty=2)
```



(2)

```
w2 = 1 / w1
wls2 = lm(sales$Sales~poly(sales$Year, 3), weight = w2)


plot(sales$Year, sales$Sales, xlab = "year", ylab = "sales", main = "WLS")
lines(sales$Year, predict(wls2), type="l", col="blue", lwd=2)

pred_interval2 <- predict(wls2, newdata = data.frame(year=sales$Year),
                          interval="prediction", level = 0.95, weight = w2)
lines(sales$Year, pred_interval2[,2], col="orange", lty=2)
lines(sales$Year, pred_interval2[,3], col="orange", lty=2)
```



(d)

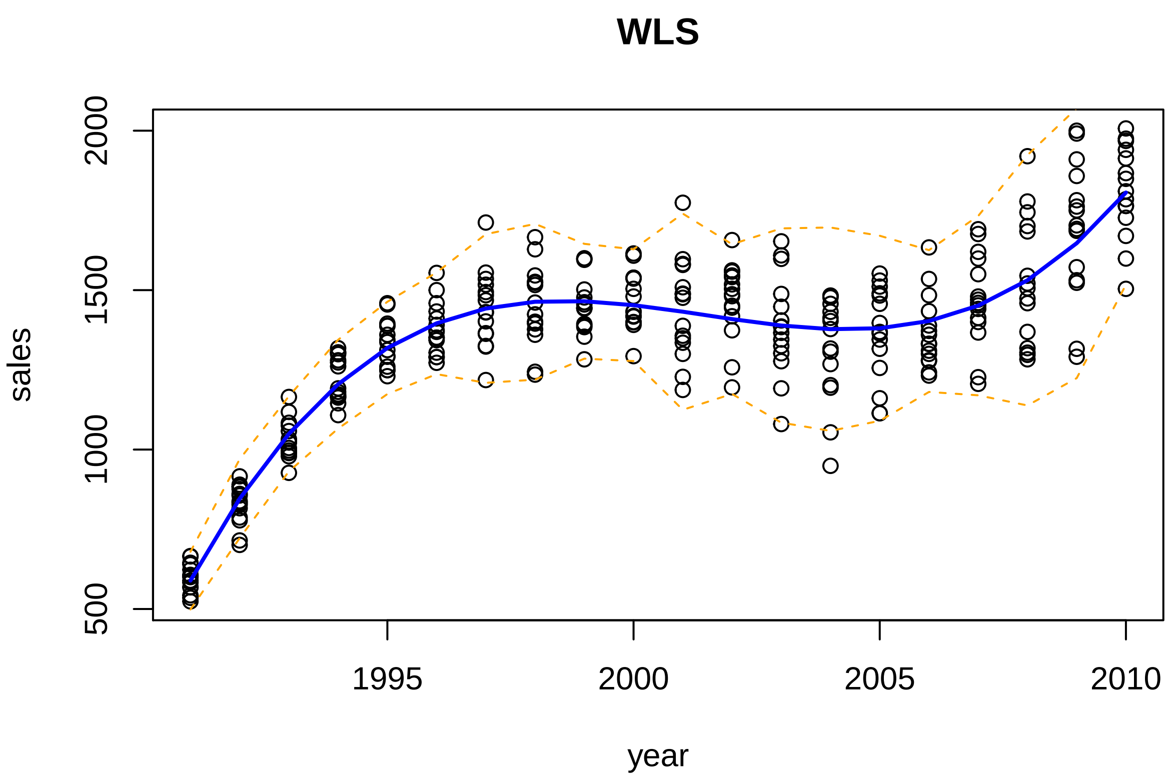
In the first model, since $\hat{\alpha}_1$ is positive, the weight increases as year increases. So the greastest influence points are when $x = 2010$. So the least influence points are when $x = 1991$.

In the second model, since $\hat{\alpha}_1$ is positive and $w = \frac{1}{\sigma^2(x)}$. The weight decreases as year increases. So the greastest influence points are when $x = 1991$. So the least influence points are when $x = 2010$.

We want to give greater weight to the points that has low variance($x = 1991$) and less weight to the points that has high variance$x = 2010$. So we prefer model 2.

q4 (a)

```
w3 = vector(length = length(sales$Year))
j = 1
for(i in sales$Year) {
  w3[j] = 1 / sigma2.y[i - 1991 + 1]
  j = j + 1
}

wls3 = lm(sales$Sales~poly(sales$Year, 3), weight = w3)

plot(sales$Year, sales$Sales, xlab = "year", ylab = "sales", main = "WLS")
lines(sales$Year, predict(wls3), type="l", col="blue", lwd=2)

pred_interval3 <- predict(wls3, newdata = data.frame(year=sales$Year),
                          interval="prediction", level = 0.95, weight = w3)
lines(sales$Year, pred_interval3[,2], col="orange", lty=2)
lines(sales$Year, pred_interval3[,3], col="orange", lty=2)
```



(b)

```r
summary(lm(sales$Sales~poly(sales$Year, 3)))
```

```
##
## Call:
## lm(formula = sales$Sales ~ poly(sales$Year, 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -423.04  -65.69   -5.62   80.61  385.48
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1353.230      7.274  186.03   <2e-16 ***
## poly(sales$Year, 3)1  3556.664    125.992   28.23   <2e-16 ***
## poly(sales$Year, 3)2 -1360.367    125.992  -10.80   <2e-16 ***
## poly(sales$Year, 3)3  2504.458    125.992   19.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 126 on 296 degrees of freedom
## Multiple R-squared:  0.8155, Adjusted R-squared:  0.8137
## F-statistic: 436.2 on 3 and 296 DF,  p-value: < 2.2e-16
```

```r
summary(wls1)
```

```
##
## Call:
## lm(formula = sales$Sales ~ poly(sales$Year, 3), weights = w1)
##
## Weighted Residuals:
##    Min     1Q Median     3Q    Max
## -63774  -7812    218   7676  65474
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1351.11      12.71 106.285  < 2e-16 ***
## poly(sales$Year, 3)1  3618.85     278.86  12.977  < 2e-16 ***
## poly(sales$Year, 3)2 -1437.25     262.23  -5.481 9.06e-08 ***
## poly(sales$Year, 3)3  2590.38     198.69  13.037  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18570 on 296 degrees of freedom
## Multiple R-squared:  0.5588, Adjusted R-squared:  0.5544
## F-statistic:   125 on 3 and 296 DF,  p-value: < 2.2e-16
```

```r
summary(wls2)
```

```
##
## Call:
## lm(formula = sales$Sales ~ poly(sales$Year, 3), weights = w2)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.3973 -0.6591 -0.0504  0.7152  3.4570
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1353.230      7.992 169.312   <2e-16 ***
## poly(sales$Year, 3)1  3556.664    138.434  25.692   <2e-16 ***
## poly(sales$Year, 3)2 -1360.367    138.434  -9.827   <2e-16 ***
## poly(sales$Year, 3)3  2384.622    105.498  22.604   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.089 on 296 degrees of freedom
## Multiple R-squared:  0.9578, Adjusted R-squared:  0.9574
## F-statistic:  2240 on 3 and 296 DF,  p-value: < 2.2e-16
```

```r
summary(wls3)
```

```
##
## Call:
## lm(formula = sales$Sales ~ poly(sales$Year, 3), weights = w3)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -2.64046 -0.64234 -0.05533  0.70605  2.27259
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1352.749      6.831  198.04   <2e-16 ***
## poly(sales$Year, 3)1  3524.757    119.961   29.38   <2e-16 ***
## poly(sales$Year, 3)2 -1397.098    118.093  -11.83   <2e-16 ***
## poly(sales$Year, 3)3  2418.002     98.417   24.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9953 on 296 degrees of freedom
## Multiple R-squared:  0.9359, Adjusted R-squared:  0.9353
## F-statistic:  1441 on 3 and 296 DF,  p-value: < 2.2e-16
```

the std. errors of the model chosen in question 2 are 7.274, 125.992, 125.992, 125.992.

the std. errors of the first model in question 3(c) are 12.71, 278.86, 262.23, 198.69.

the std. errors of the second model in question 3(c) are 7.992, 138.434, 138.434, 105.498.

the std. errors of the model in question 4(a) are 6.831, 119.961, 118.093, 98.417.

The model in 4(a) has the lowest std. error among all parameters. I will choose the model in 4(a).

q5 Question: Is the minimum value of MSE unique(i.e. there exits two different complexicities that MSE are minimum)?

Answer: True.

Explanation: $MSE = [Bias(\hat{f}(x_0))]^2 + Var(\hat{f}(x_0)) + Var(\epsilon)$. The bias decreases as complexity increases, since it becomes more and more close to the value of y. In addition, the variance increases as complexity increases, since the model follows the error/noise too closely. Since $Var(\epsilon)$ is constant, there minimum value of MSE is unique.