a1

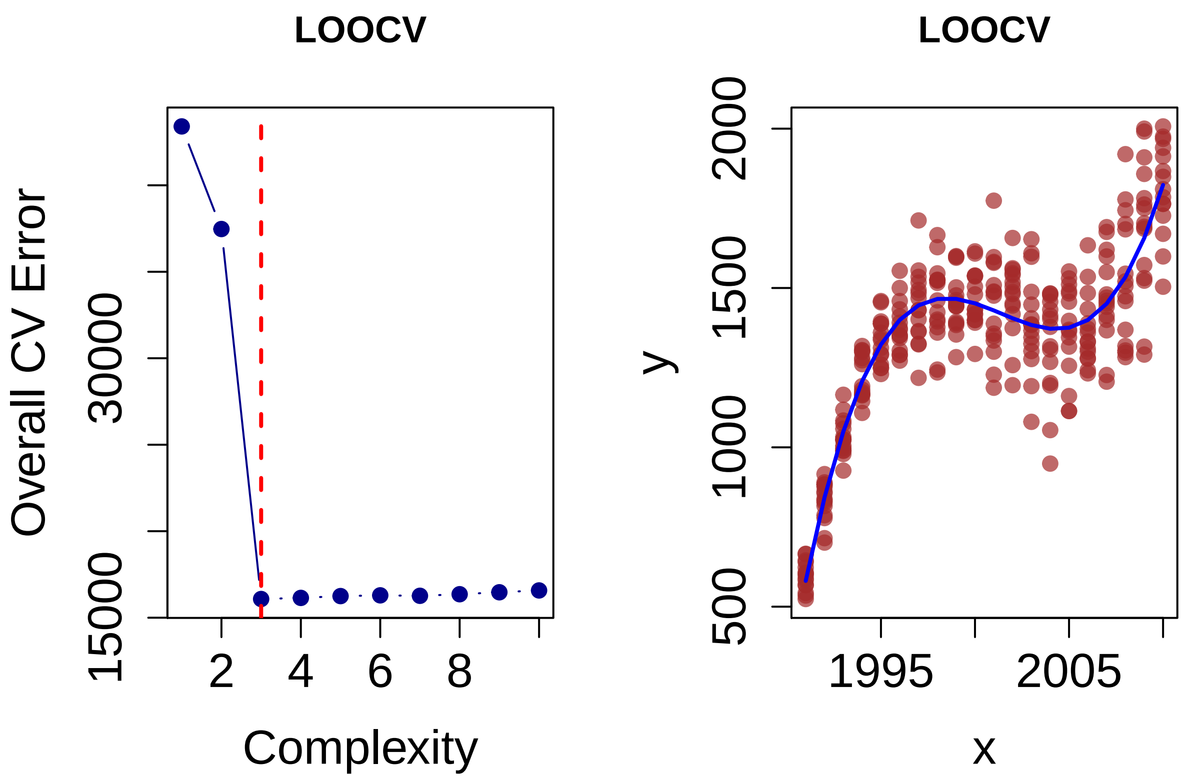
2(a)

```r
# 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)
                   ]
         }
  )
```

```r
}


# A function to form the k samples
getKfoldSamples <- function (x, y, k, indices=NULL){
  groups <- kfold(length(x), k, indices)
  Ssamples <- lapply(groups,
                     FUN=function(group) {
                       list(x=x[-group], y=y[-group])
                     })
  Tsamples <- lapply(groups,
                     FUN=function(group) {
                       list(x=x[group], y=y[group])
                     })
  list(Ssamples = Ssamples, Tsamples = Tsamples)
}
```

```r
sales <- read.table(file="JaxSales.txt", header=T)
x=sales$Year
y=sales$Sales

# For leave one out cross-validation
samples_loocv <-  getKfoldSamples(x, y, k=length(y))
# 10 fold cross-validation
samples_10fold <-  getKfoldSamples(x, y, k=10)

complexity <- c(1:10) # Thiese are the degrees of polynomials to be fitted

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))
}

par(mfrow=c(1,2))

Title.Graph = "LOOCV" # 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)

plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5)
lines(x, predict(lm(y~poly(x,indx))), type="l", col="blue", lwd=2)
```

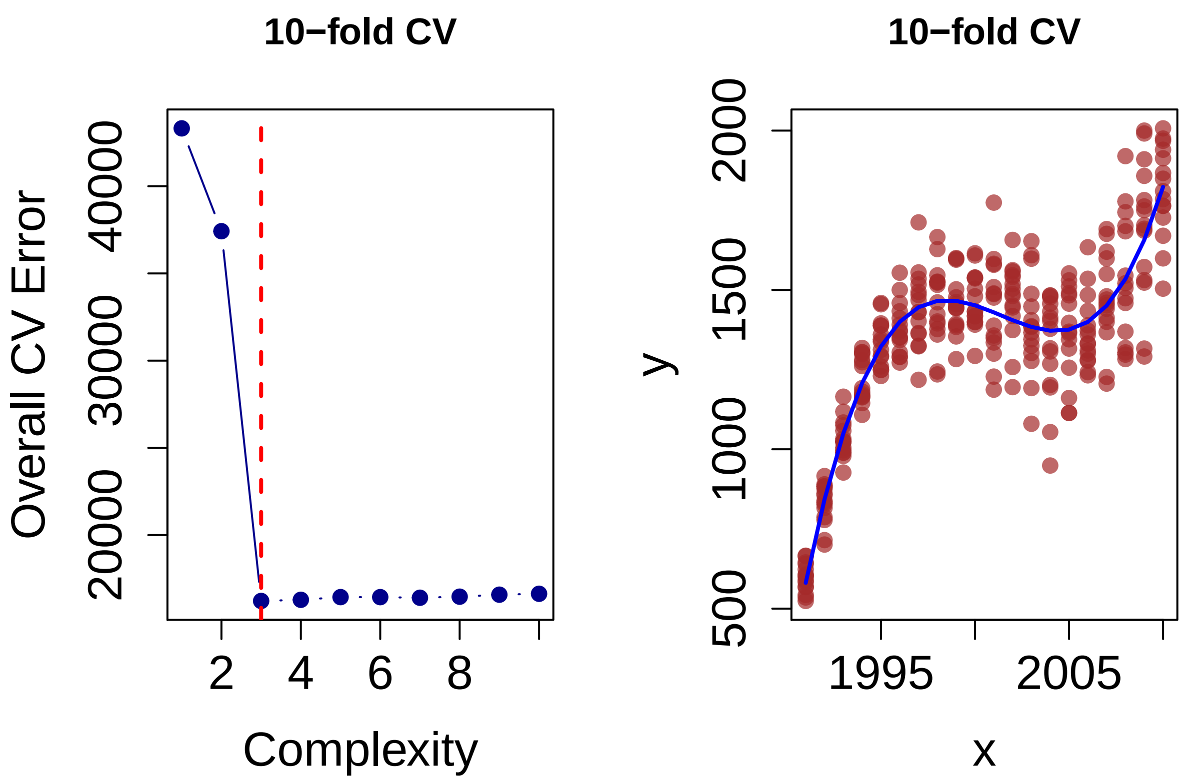
```
reg_q2 = lm(y~poly(x,3))
text(8,7,"Black: True")
text(8,6.5,"Blue: Fitted" , col="blue")
title(main=Title.Graph)
```



2(b)

```
sales <- read.table(file="JaxSales.txt", header=T)
x=sales$Year
y=sales$Sales


complexity <- c(1:10) # Thiese are the degrees of polynomials to be fitted

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)
  }
```

```r
  CV.To.Plot[i,] = c(complexity[i] , mean(MSE))
}

par(mfrow=c(1,2))

Title.Graph = "10-fold 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)

plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5)
lines(x, predict(lm(y~poly(x,indx))), type="l", col="blue", lwd=2)
text(8,7,"Black: True")
text(8,6.5,"Blue: Fitted" , col="blue")
title(main=Title.Graph)
```

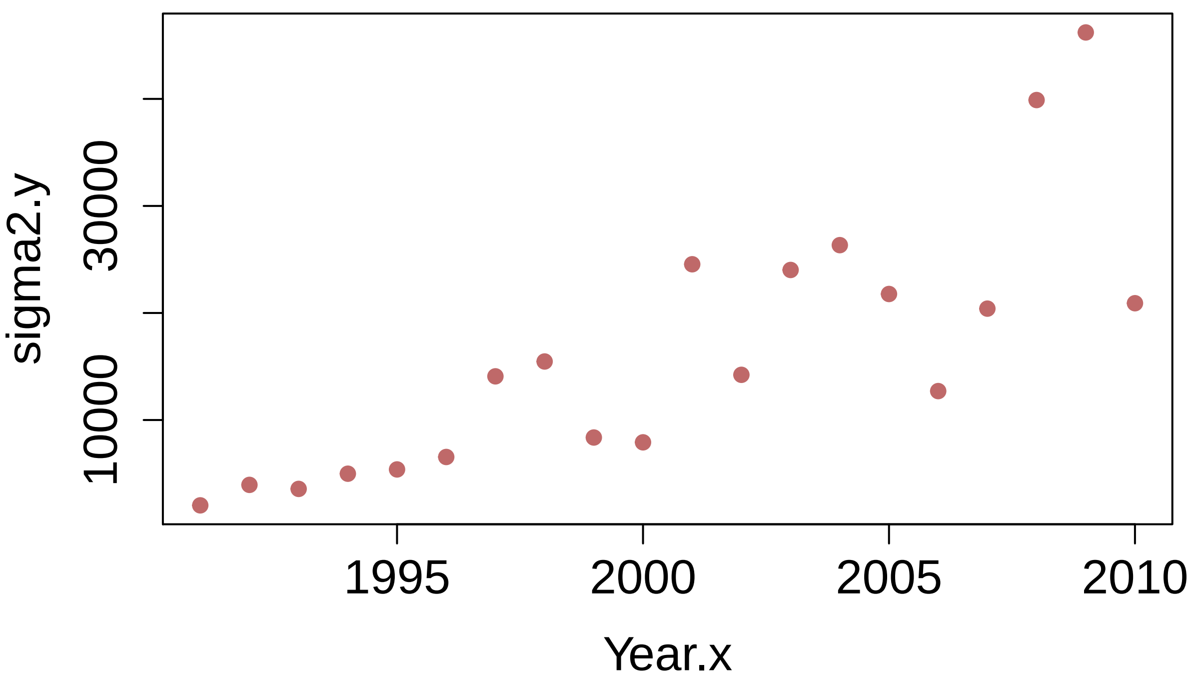


2(c)

they result in the same model (complexity=3). I prefer when k=10 because firstly it's more efficient for my code (k=n could be more effiecient if using other algorithm but I did not use it) and secondly k=10 result in less variance because the training set have less 'curse of linearity'

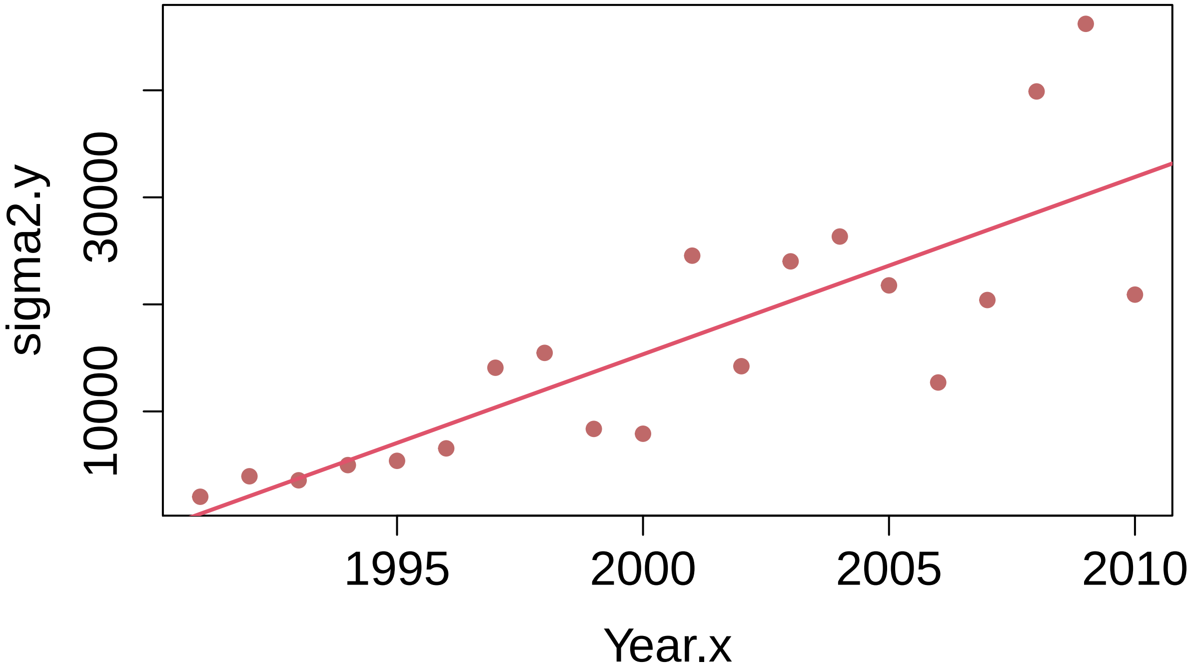
3(a)

```
Year.x=c(1991:2010)
indxs=c()
for (i in 1:length(Year.x)){
  indxs[[i]]=which(sales$Year==Year.x[i])
}

sigma2.y=sapply(Year.x,
       FUN=function(i) {
         indx = which(sales$Year==i)
         data = sales$Sales[indx]
         data.var = var(data)
         data.var
       }
)

plot(sigma2.y~Year.x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5)
```



3(b)

```
plot(sigma2.y~Year.x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5)

reg = lm( sigma2.y ~ Year.x)
abline(reg , col=2 , lwd=2)
```




```
reg$coefficients
```

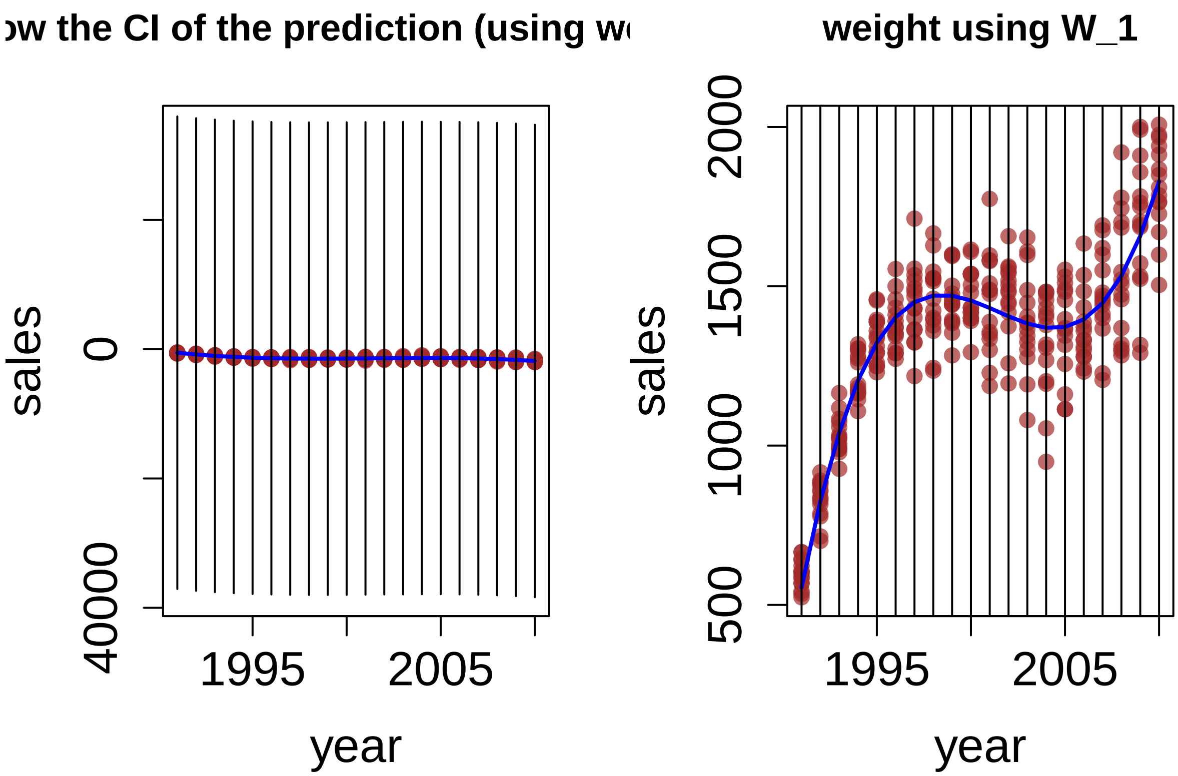
```
##  (Intercept)       Year.x
## -3297581.598     1656.459
```

```
alpha_0 = reg$coefficients[1]
alpha_1 = reg$coefficients[2]
```

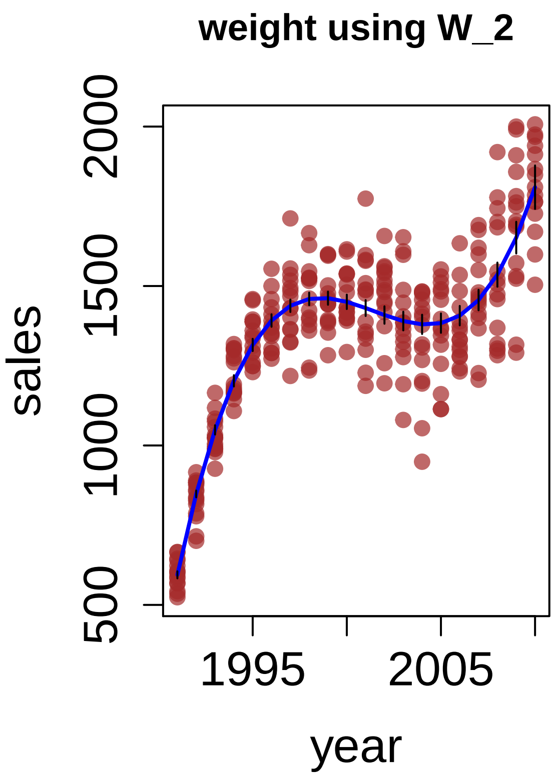
3(c)

```
W_1 = (alpha_0 + alpha_1 * sales$Year)
y=sales$Sales
x=sales$Year
weighted.LS = lm(y~poly(x , 3), weights=W_1)
newd=data.frame(x=c(1991:2010))
prediction = predict(weighted.LS, newdata = newd, interval = "prediction", level=0.95, weights = 1)
indx.max=which.max(prediction[,3])
indx.min=which.max(prediction[,2])
reg_q3_1 = weighted.LS
par(mfrow=c(1,2))

plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5, ylim=c(prediction[,3][indx
     ,xlab='year', ylab='sales', main='To show the CI of the prediction (using weight W_1)')

lines(x, predict(weighted.LS), type="l", col="blue", lwd=2)
segments(x0=Year.x, x1=Year.x, y0=prediction[,3], y1=prediction[,2])

plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5,xlab='year', ylab='sales', m
```

```
lines(x, predict(weighted.LS), type="l", col="blue", lwd=2)
segments(x0=Year.x, x1=Year.x, y0=prediction[,3], y1=prediction[,2])
```




```
W_2 = 1/(alpha_0 + alpha_1 * sales$Year)
weighted.LS = lm(y~poly(x, 3), weights=W_2)
plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5,xlab='year', ylab='sales', 
lines(x, predict(weighted.LS), type="l", col="blue", lwd=2)
newd=data.frame(x=c(1991:2010))
prediction = predict(weighted.LS, newdata = newd, interval = "prediction", level=0.95, weights = 1)
segments(x0=Year.x, x1=Year.x, y0=prediction[,3], y1=prediction[,2])
reg_q3_2 = weighted.LS
```



3(d) for the regression curce, two weights generate similar result, W_1 has slightly more variance. the 95% prediction interval of W_1 is significantly larger.

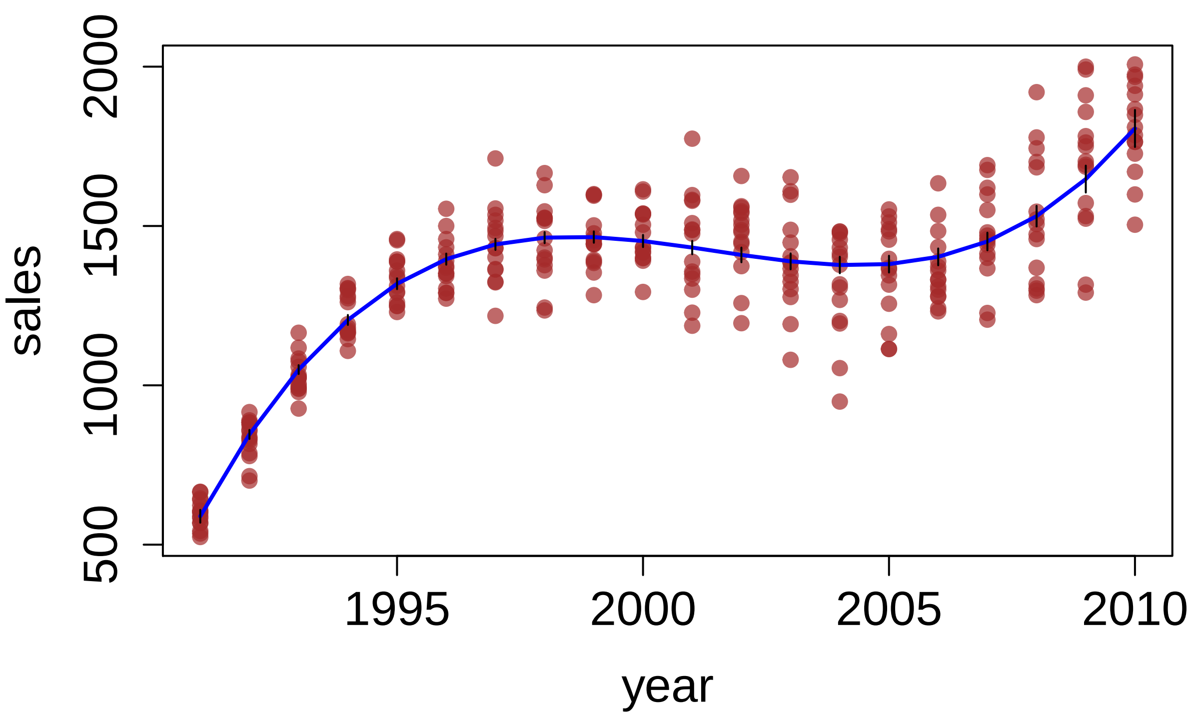
For (1), points which are larger in year have more influnce to the model. moreover points have more influnce if they have more absolute vertical distance from the fitted curve points which are smaller in year have smaller influnce to the model. moreover points have less influnce if they have smaller absolute vertical distance from the fitted curve

For (2), points which are smaller in year have more influnce to the line. moreover points have more influnce if they have more absolute vertical distance from the fitted curve points which are larger in year have smaller influnce to the model. moreover points have less influnce if they have small absolute vertical distance from the fitted curve

I would choose (2) because the larger the variance at X=x, the smaller the weight of data at X=x should be. In this case, the variance are larger for bigger value of X. therefor bigger value of X should have less influnce. Also the prediction power of (2) is significant better than prediction power of (1)

4(a)

```
W=sapply(x,
       FUN=function(i) {
         indx = which(Year.x==i)
          temp = sigma2.y[indx]
          1/temp
       }
)

y=sales$Sales
x=sales$Year

weighted.LS = lm(y~poly(x , 3), weights=W)
newd=data.frame(x=c(1991:2010))
prediction = predict(weighted.LS, newdata = newd, interval = "prediction", level=0.95, weights = 1)

plot(y~x,pch=19 , col=adjustcolor("brown",0.7) , cex.axis=1.5 , cex.lab=1.5
     ,xlab='year', ylab='sales')

lines(x, predict(weighted.LS), type="l", col="blue", lwd=2)
segments(x0=Year.x, x1=Year.x, y0=prediction[,3], y1=prediction[,2])
```




```
reg_q4 = weighted.LS
```

4(b)

```r
print('the standard error for parameters in q2 are')
```

```
## [1] "the standard error for parameters in q2 are"
```

```r
sqrt(diag(vcov(reg_q2)))
```

```
## (Intercept) poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
##    7.274157  125.992099  125.992099  125.992099
```

```r
print('the standard error for parameters in q3/1 are')
```

```
## [1] "the standard error for parameters in q3/1 are"
```

```r
sqrt(diag(vcov(reg_q3_1)))
```

```
## (Intercept) poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
##    12.71207   278.86060   262.22599   198.69143
```

```r
print('the standard error for parameters in q3/2 are')
```

```
## [1] "the standard error for parameters in q3/2 are"
```

```r
sqrt(diag(vcov(reg_q3_2)))
```

```
## (Intercept) poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
##    7.992499  138.434146  138.434146  105.497764
```

```r
print('the standard error for parameters in q4 are')
```

```
## [1] "the standard error for parameters in q4 are"
```

```r
sqrt(diag(vcov(reg_q4)))
```

```
## (Intercept) poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
##    6.830807  119.960741  118.092695   98.416831
```

We observe that q4 have smallest variance, q1 and q3/2 have similar variance, q3/1 has largest variance. We would choose the smallest variance (q4) because smaller variance means higher prediction power, smaller 95% confidence interval range.