Project-5

Chitra Karki* University of Texas at El Paso (UTEP)

November 08, 2022

Contents

We consider a data set jaws.txt, which is concerned about the association between jaw bone length (y = bone) and age in deer (x = age). We are going try out several parametric/nonparametric nonlinear regression models in this low-dimensional (p = 1) setting.

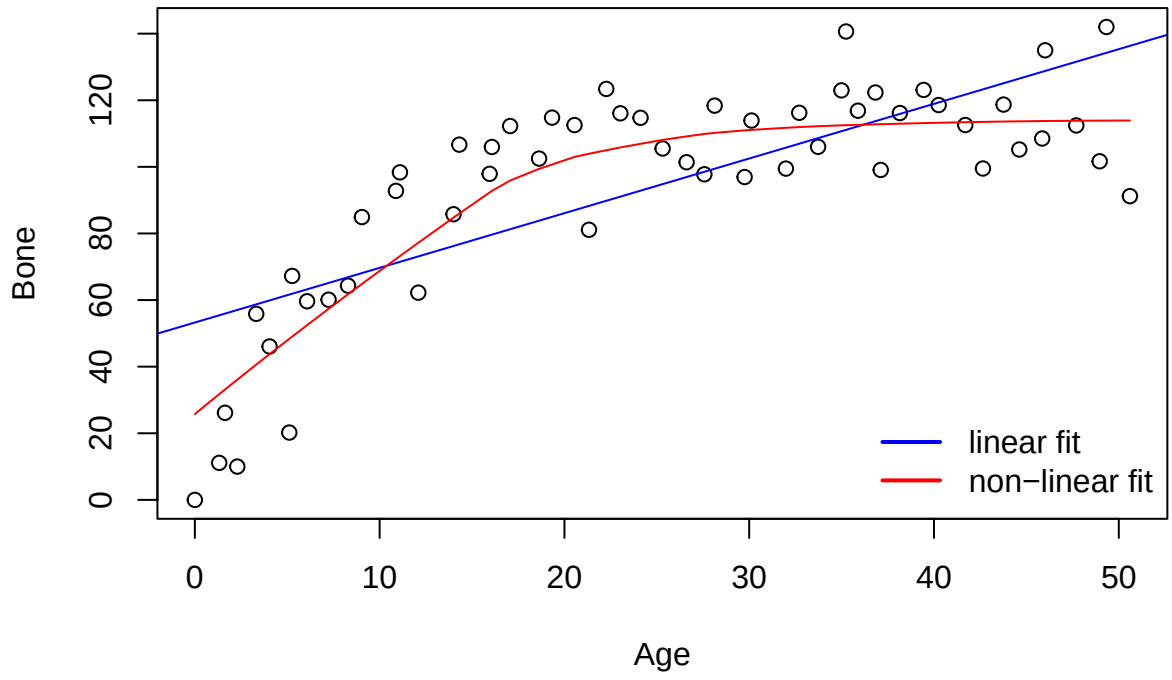
1 Reading Data

Bring in the data D and make a scatterplot of bone vs. age. Optionally, add a linear fit and a nonlinear fit (with, e.g., lowess or loess) and inspect their discrepancy. Does their association look linear?

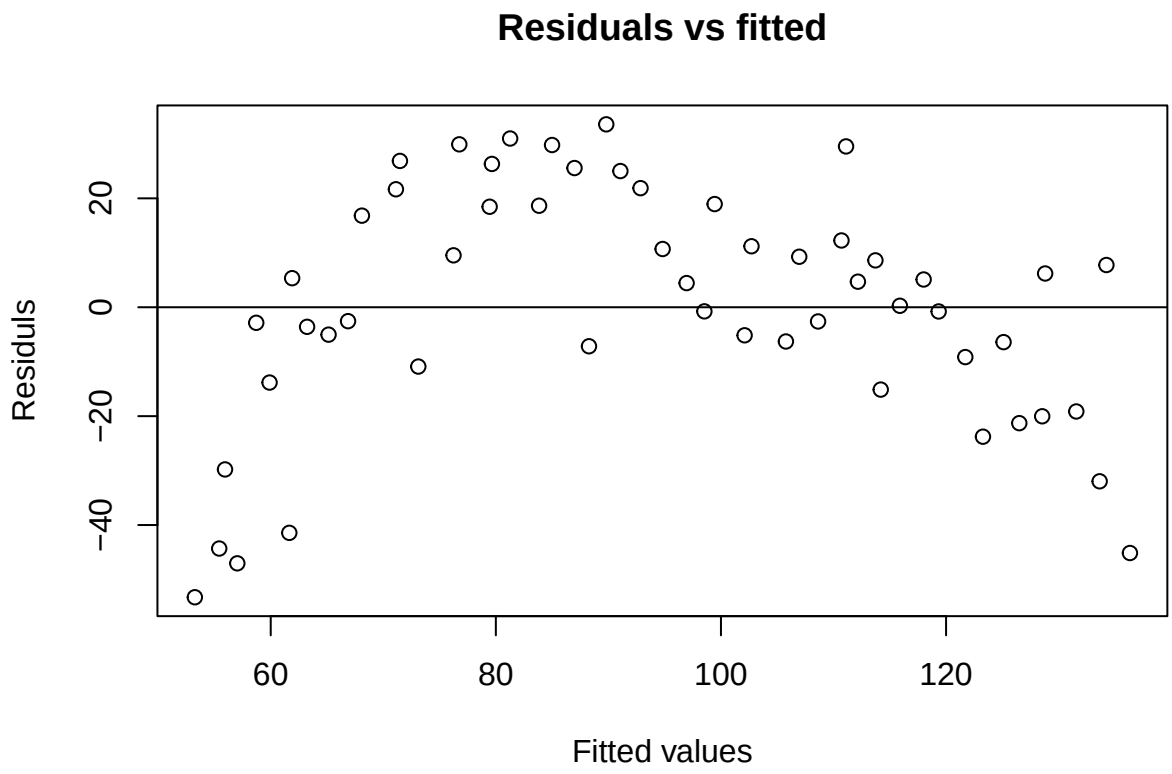
```
setwd("C:/Users/chitr/OneDrive - University of Texas at El Paso/data_science/semesters/sem3-fa

D = read.table("jaws.txt",header = T)
```

*cbkarki@miners.utep.edu

```
# plot bone Vs age

lin.fit = lm(bone~age,data = D)
plot(y=D$bone,x=D$age,xlab = "Age",ylab="Bone")
abline(lin.fit,col="blue")
# non-linear (lowess/loess)
lines(lowess(y=D$bone,x=D$age),col="red")
legend('bottomright',
       col = c('blue', 'red'),
       lwd = 2,
       c('linear fit', 'non-linear fit'),bty = "n")
```




```
plot(y = lin.fit$residuals,x=lin.fit$fitted.values,
     xlab="Fitted values",ylab = "Residuls",main = "Residuals vs fitted")
abline(h=0)
```



Comment: The association does not look linear. If we look at the linear fit, its observed that the fitted line do not fit well with the data points.The lowess fit with the default setting shows its polynomial with some degree which somehow follow the pattern of the data points. Also, the residuals vs fitted values plots shows the linear fit didn't do a good job, the pattern is non-linear.

2 Data partitioning

(a) Randomly partition the data D into the training set D1 and the test set D2 with a ratio of approximately 2:1 on the sample size.

(b) To prevent extrapolation when it comes to prediction, the range of age in the test set D2 should not exceed that in the training set D1: Find the (two) observations with minimum and maximum age in data D and force them to go to the training set D1 if they are not in D1.

```
# 2(a)
# partitioning data
set.seed(123)
d1.index = sample(1:nrow(D),size =(2/3)*nrow(D))
if (!which(D$age==max(D$age)) %in% d1.index){
        d1.index = c(d1.index,which(D$age==max(D$age)))
}
```

```r
if (!which(D$age==min(D$age)) %in% d1.index) {
        d1.index = c(d1.index,which(D$age==min(D$age)))

}
D1 = D[d1.index,]
D2 = D[-d1.index,]

# ordering based on age
D1 = D1[order(D1$age),]
D2 = D2[order(D2$age),]
age = D1[,1]
bone = D1[,2]

# 2(b)
# checking min max
# range(D$age);range(D1$age);range(D2$age)
#
# sum(D$age==max(D$age)) # the occurance of max age is only once
# sum(D$age==min(D$age))
#
# # forcing observation  with max and min age of D if present in D2 to D1
# d1.index = c(d1.index,which(D$age==max(D$age) | D$age==min(D$age)))
# D1 = D[d1.index,]
# D2 = D[-d1.index,]
#
# # checking if its ok now
range(D$age);range(D1$age);range(D2$age)
```

```
## [1]  0.0000 50.6041
```

```
## [1]  0.0000 50.6041
```

```
## [1]  2.297635 46.015332
```

Comment: Data set was randomly splited into training and the testing sets. The observation having min and the max ages were forced to move in training set.

3 Parametric non-linear models

First consider parametric nonlinear models.

(a) Fit an asymptotic exponential model of the following form

$$y = \beta_1 - \beta_2 e^{-\beta_3 x} + \epsilon \tag{1}$$

with the training set D1: Provide a summary of the fitted model and interpret the results.

(b) To test $H_0 : \beta_1 = \beta_2$ in Model (1), fit the reduced model, i.e., the model by plugging in the condition under H_0 and use the anova function. Also, compare two **nls** models with AIC/BIC. Then conclude on which model is better.

(c) Based on the better model in 2(b), add the fitted curve to the scatterplot.

(d) Apply the better model in 2(b) to the test set D2. Plot the observed y_i values in D2 versus their predicted values $\hat{y}_i$, together with the reference line $y = x$, to check if the prediction seems reasonable. And computer the prediction mean square error (MSE)

$$MSE = \frac{1}{|D|} \sum_{i \in D_2} (y_i - \hat{y}_i)$$

```
# (a)

#testing for initialization for which the pattern of the scatter plot is matched somehow.
# asy.exp = function(x,beta1,beta2,beta3){
#   y = beta1 - beta2*(exp(-beta3*x))
# }
#
#
# plot(asy.exp(1:40,1,1,0.12))
#attach(D1)
jaw.mod <- nls(bone ~ beta1 - beta2*(exp(-beta3*age)),
    start=list(beta1 = 1, beta2 = 1, beta3 =0.12), trace=T)
```

```
## 356317.7    (8.07e+00): par = (1 1 0.12)
## 38693.92    (1.11e+00): par = (112.5346 110.9153 1.682318)
## 13454.75    (1.06e+00): par = (109.274 107.9583 0.3335579)
## 6216.459    (3.86e-01): par = (108.9916 105.0474 0.1748338)
## 5454.669    (1.13e-01): par = (112.0298 110.6171 0.1295915)
## 5385.844    (4.02e-03): par = (112.6142 111.2875 0.1354257)
## 5385.754    (1.84e-04): par = (112.6066 111.3891 0.1358514)
## 5385.754    (9.75e-06): par = (112.6047 111.3934 0.1358744)
```

```
summary(jaw.mod)
```

```
##
## Formula: bone ~ beta1 - beta2 * (exp(-beta3 * age))
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 112.60475    3.04181  37.019  < 2e-16 ***
## beta2 111.39343    8.43085  13.213 3.65e-15 ***
## beta3   0.13587    0.02122   6.404 2.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 12.4 on 35 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 9.747e-06
```

Comment: The coefficients were initialized at [1,1,0.12] for beta1,beta2 and beta3 respectively. The initialization values were obtained from some hit and trails,specially on beta3. They converged to the values, [120.1008 108.4289 0.08832346] in 10 iterations. From summary of the model,the coefficients seems significant.

```
#(b)

# asy.exp = function(x,beta1,beta3){
#    y = beta1 - beta1*(exp(-beta3*x))
#  }
# plot(asy.exp(1:40,0.10,0.12))

jaw.mod.red <- nls(bone ~ beta1*(1-(exp(-beta3*age))),
    start=list(beta1 = 1, beta3 =0.12), trace=T)
```

```
## 356317.7    (8.06e+00): par = (1 0.12)
## 39566.58    (1.11e+00): par = (112.3644 1.957549)
## 30453.49    (1.08e+00): par = (110.7239 0.9584042)
## 20021.62    (1.09e+00): par = (108.3748 0.5286904)
## 6345.210    (4.10e-01): par = (106.7847 0.1839043)
## 5490.204    (1.36e-01): par = (111.7698 0.1306862)
## 5388.855    (4.74e-03): par = (112.5018 0.1375465)
## 5388.732    (1.36e-04): par = (112.5028 0.137899)
## 5388.732    (4.85e-06): par = (112.5015 0.137912)
```

```
summary(jaw.mod.red)
```

```
##
## Formula: bone ~ beta1 * (1 - (exp(-beta3 * age)))
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 112.50147    2.90106  38.779  < 2e-16 ***
## beta3   0.13791    0.01583   8.714 2.15e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 36 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 4.852e-06
```

The model was reduced as suggested by the null hypothesis. The modeling non-linear function is now, $\hat{y} = \beta_1(1 - e^{\beta_3 x})$ with two coefficients. The coefficients converged to the values (117.0201 0.1096543) for beta1 and beta2 respectively in 8 iterations. Summary of the reduced model shows, the coefficients are significant.

```
# comparison of models

# anova
anova(jaw.mod,jaw.mod.red)
```

```
## Analysis of Variance Table
##
## Model 1: bone ~ beta1 - beta2 * (exp(-beta3 * age))
## Model 2: bone ~ beta1 * (1 - (exp(-beta3 * age)))
##   Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
## 1     35     5385.8
## 2     36     5388.7 -1 -2.9778  0.0194 0.8902
```

```
# AIC/BIC
AIC(jaw.mod);AIC(jaw.mod.red)
```

```
## [1] 304.0885
```

```
## [1] 302.1095
```

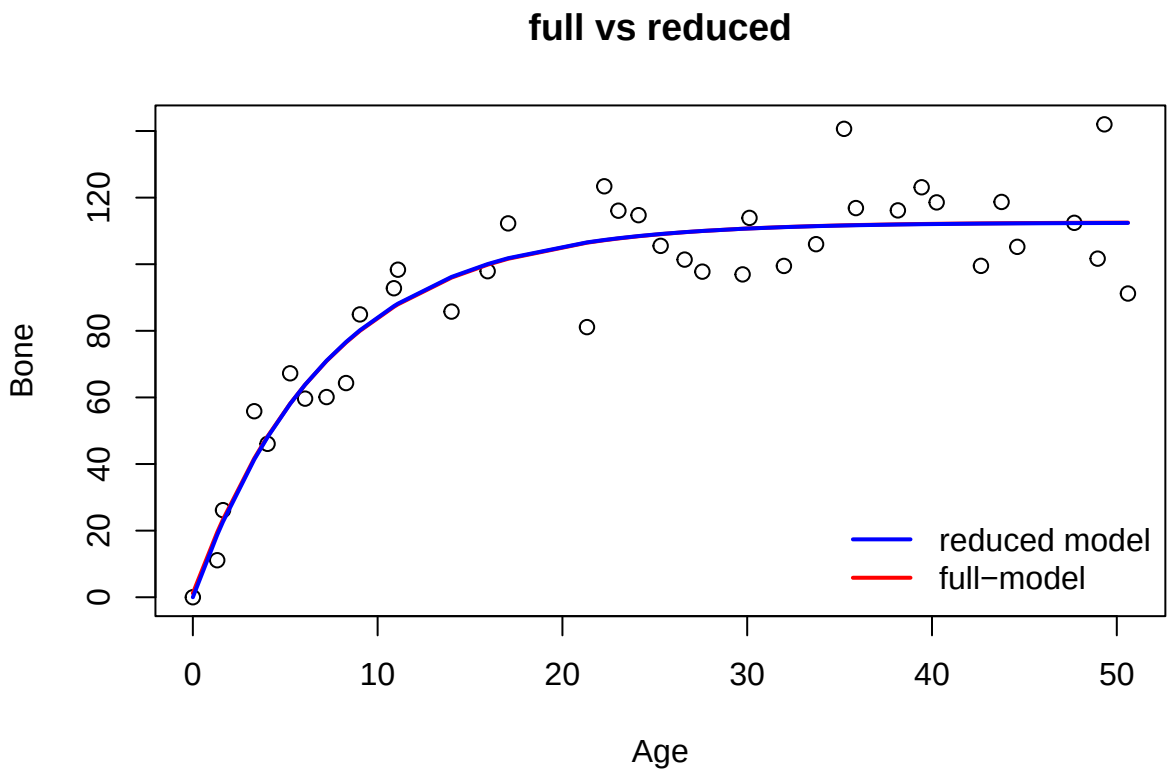
```
BIC(jaw.mod);BIC(jaw.mod.red)
```

```
## [1] 310.6389
```

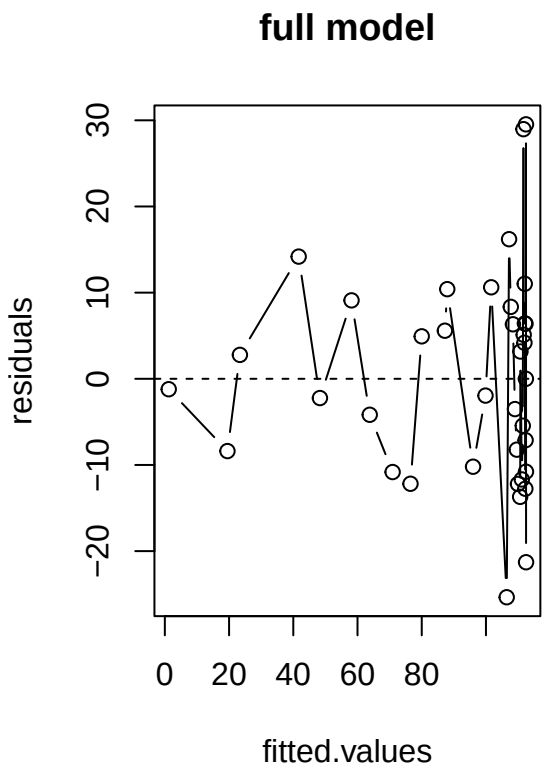
```
## [1] 307.0223
```

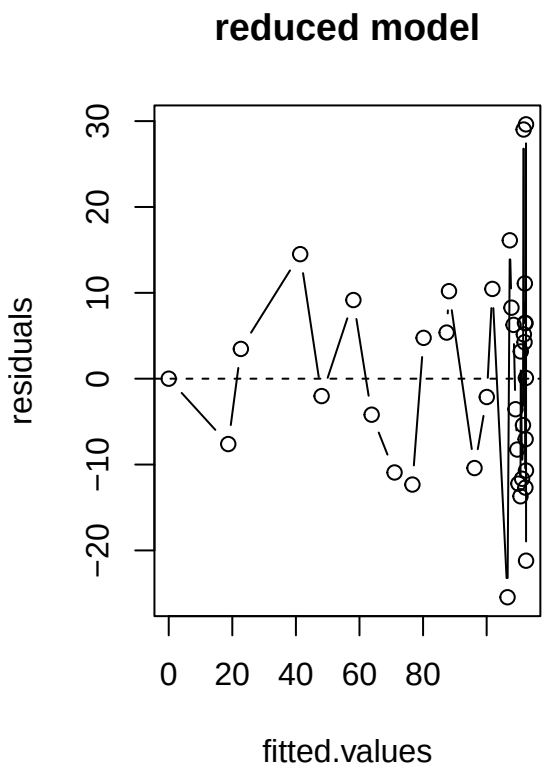
Comment: on comparing these 2 models, AIC/BIC and MSE for jaw.mod.red has lesser values then jaw.mod model. Hence, based on these criterion, reduced model is better. But, these two models are comparable, results are close to each other.

```
#plots fitted curve
par(mfrow=c(1,1))
plot(y=bone,x=age,xlab = "Age",ylab="Bone",main = "full vs reduced")
lines(x=age, y=fitted.values(jaw.mod), lwd=2, col="red")
lines(x=age, y=fitted.values(jaw.mod.red), lwd=2, col="blue")
legend('bottomright',
       col = c('blue', 'red'),
       lwd = 2,
       c('reduced model', 'full-model'),bty = "n")
```




```
# plots of residuals
par(mfrow=c(1,2))
plot(x=fitted.values(jaw.mod), y=residuals(jaw.mod), type='b',
     main = "full model",xlab = "fitted.values",ylab = "residuals")
abline(h=0, lty=2)

plot(x=fitted.values(jaw.mod.red), y=residuals(jaw.mod.red), type='b',
     main = "reduced model",xlab = "fitted.values",ylab = "residuals")
abline(h=0, lty=2)
```

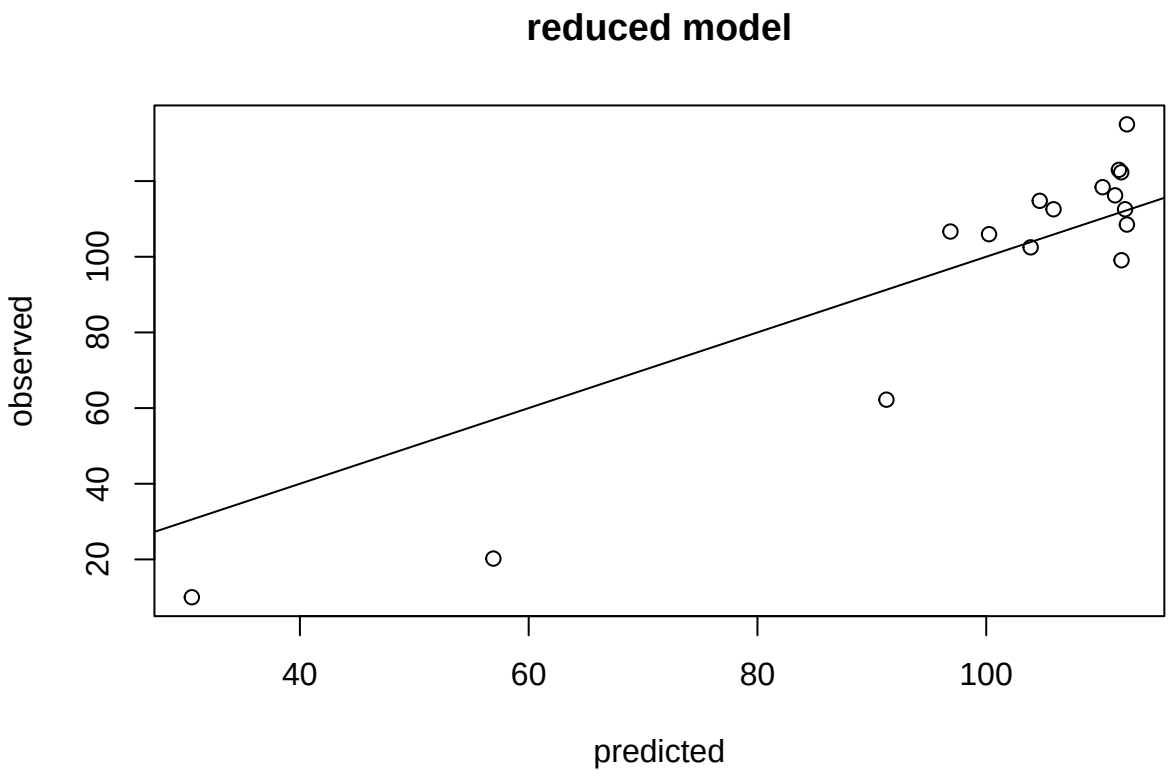





```
par(mfrow=c(1,1))
```

Comment: The fitted curves for both the models seems to follow the pattern of the data points. The fitted curves for both the models are difficult to distinguish, as there is a small difference in the models. The residuals vs fitted values shows patters. For good models the distribution of the residuals should be random with respect to the fitted curve.

```
# (d)
#prediction on model jaw.mod.red

y.hat = predict(jaw.mod.red,newdata = data.frame(age=D2$age))
plot(x = y.hat,y=D2$bone,xlab = "predicted",ylab = "observed",main = "reduced model")
abline(a=0,b=1)
```




```
# MSE
# MSE for reduced model
jaw.mod.red.MSE = mean((bone - predict(jaw.mod.red))^2);jaw.mod.red.MSE
```

```
## [1] 141.8087
```

Comment: The observed vs predicted values plot for the model jaw.mod.red shows that the points are not in close agreement with y=x line. so it didn't performed well.And MSE is 141.8087.

4 Local regression methods

Next consider local regression methods.

(a) On basis of D1; obtain a KNN regression model with your choice of K. Plot the fitted curve together with the scatterplot of the data. Apply the fitted model to D2. Plot the observed and predicted response values with reference line y = x and obtain the prediction MSE.

(b) Apply kernel regression to obtain a nonlinear fit. State your choice of the kernel function and the choice of your bandwidth. Explain how you decide on the choices of kernel and bandwidth. Apply the fitted kernel regression model to the test data D2. Plot the observed and predicted response values with reference line y = x and obtain the prediction MSE.

(c) Apply local (cubic) polynomial regression to the training data D1: Again, state your choice of the kernel function and the bandwidth used. Apply the local cubic regression model to the test data D2. Plot the observed and predicted response values with reference line y = x and obtain the prediction MSE.

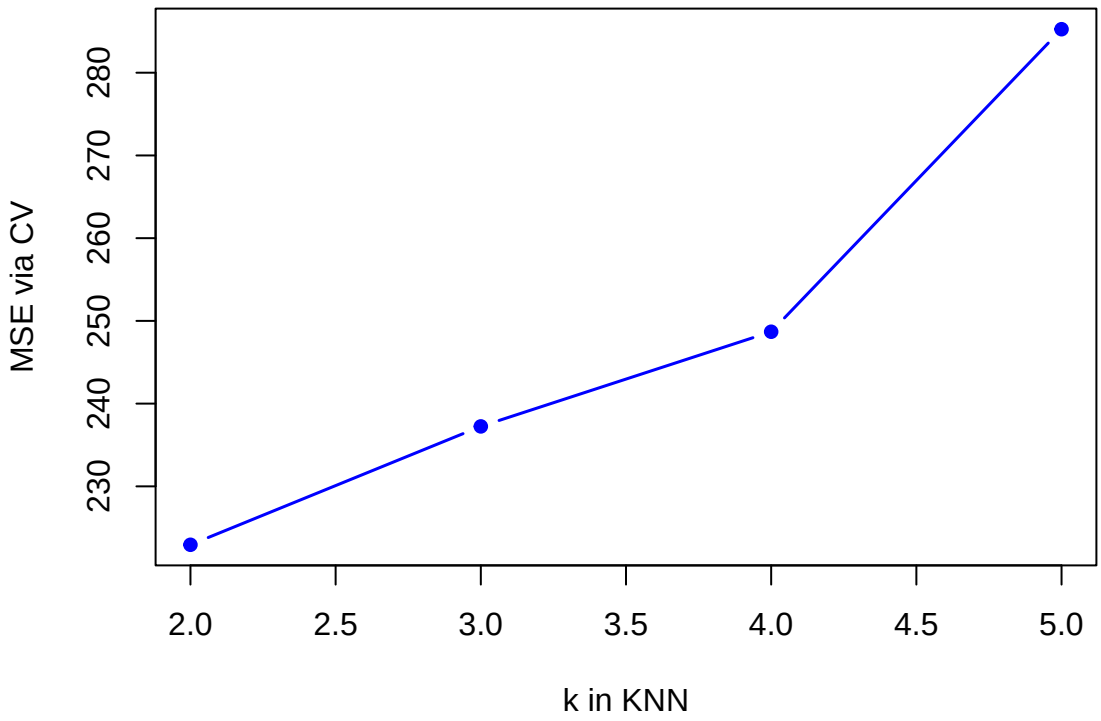
```
# (a)
# V-FOLD CV FOR SELECTING K
library("FNN")
```

```
## Warning: package 'FNN' was built under R version 4.2.2
```

```
set.seed(123)
V <- 3; n <- NROW(D1)
id.fold <- sample(x=1:V, size=n, replace=TRUE)
K <- 2:5; SSE <- rep(0, length(K))
for (k in 1:length(K)){
    for (v in 1:V){
        train.v <- D1[id.fold!=v,];
        test.v <- D1[id.fold==v,]
        yhat <- knn.reg(train=train.v, test=test.v, y=train.v$bone, k=K[k])$pred;
        SSE[k] <- SSE[k] + sum((test.v$bone-yhat)^2)
    }
}
cbind(K, SSE/n)
```

```
##      K
## [1,] 2 222.9371
## [2,] 3 237.2443
## [3,] 4 248.6867
## [4,] 5 285.2575
```

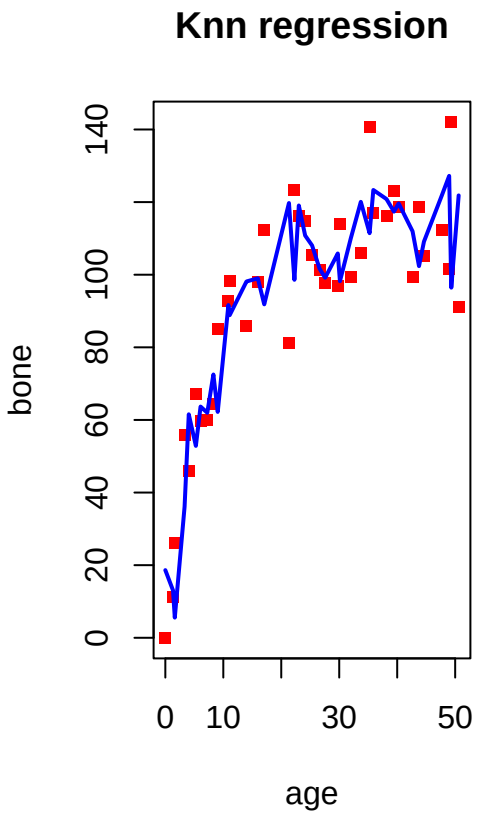
```
par(mfrow=c(1, 1), mar=rep(4,4))
plot(K, SSE/n, col="blue", pch=19, cex=.8, type="b", xlab='k in KNN',
    ylab="MSE via CV", lwd=1.5)
```

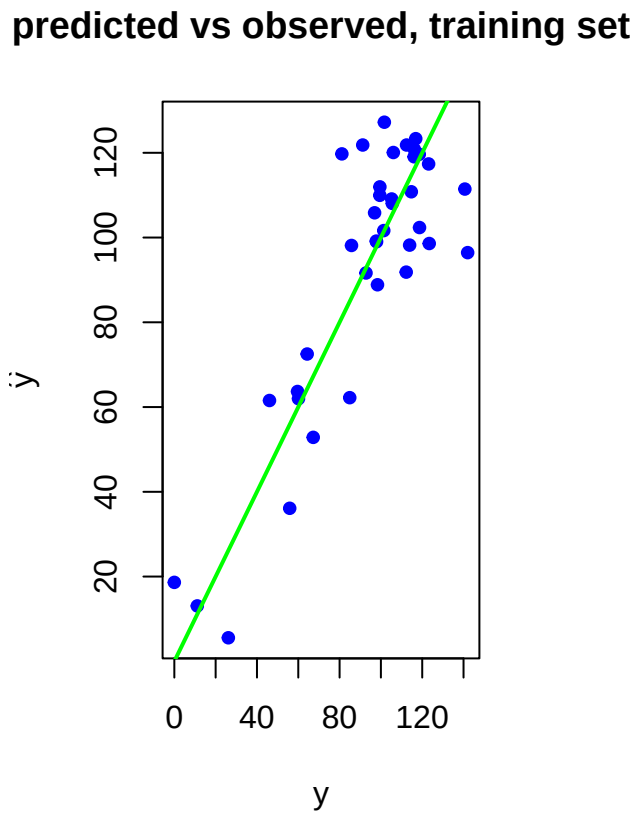



```
k.opt <- K[which.min(SSE)] ;k.opt
```

```
## [1] 2
```

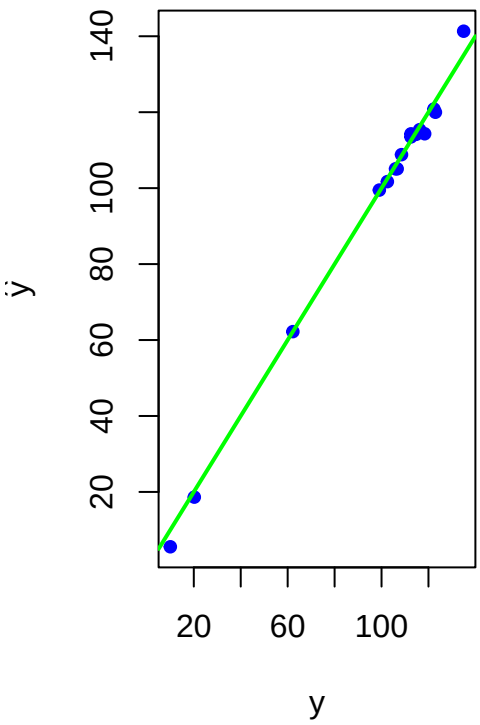
```
# fitting knn with optimal K neighbours
fit.knn <- knn.reg(train=age, y=bone, k=k.opt, algorithm="kd_tree");
par(mfrow=c(1, 2), mar=rep(4,4))
plot(y=bone, x=age, xlab="age", ylab="bone",
    col="red", pch=15, cex=0.8,main="Knn regression")
lines(x=age, y=fit.knn$pred, col="blue", lwd=2)
# PREDICTED VS. OBSERVED
plot(x=bone, y=fit.knn$pred, col="blue", pch=19,
    xlab="y", ylab=expression(hat(y)), cex=0.8,main="predicted vs observed, training set")
abline(a=0, b=1, col="green", lty=1, lwd=2)
```






```
# MAKING PREDICTION on test data D2
fit.knn <- knn.reg(train=D1, test=D2, y=D1$bone, k=k.opt, algorithm="kd_tree");
plot(x=D2$bone, y=fit.knn$pred, col="blue", pch=19,
    xlab="y", ylab=expression(hat(y)), cex=0.8,main="predicted vs observed, test set")
abline(a=0, b=1, col="green", lty=1, lwd=2)

# prediction MSE
fit.knn.MSE = mean((fit.knn$pred-D2$bone)^2);fit.knn.MSE
```

```
## [1] 6.197896
```

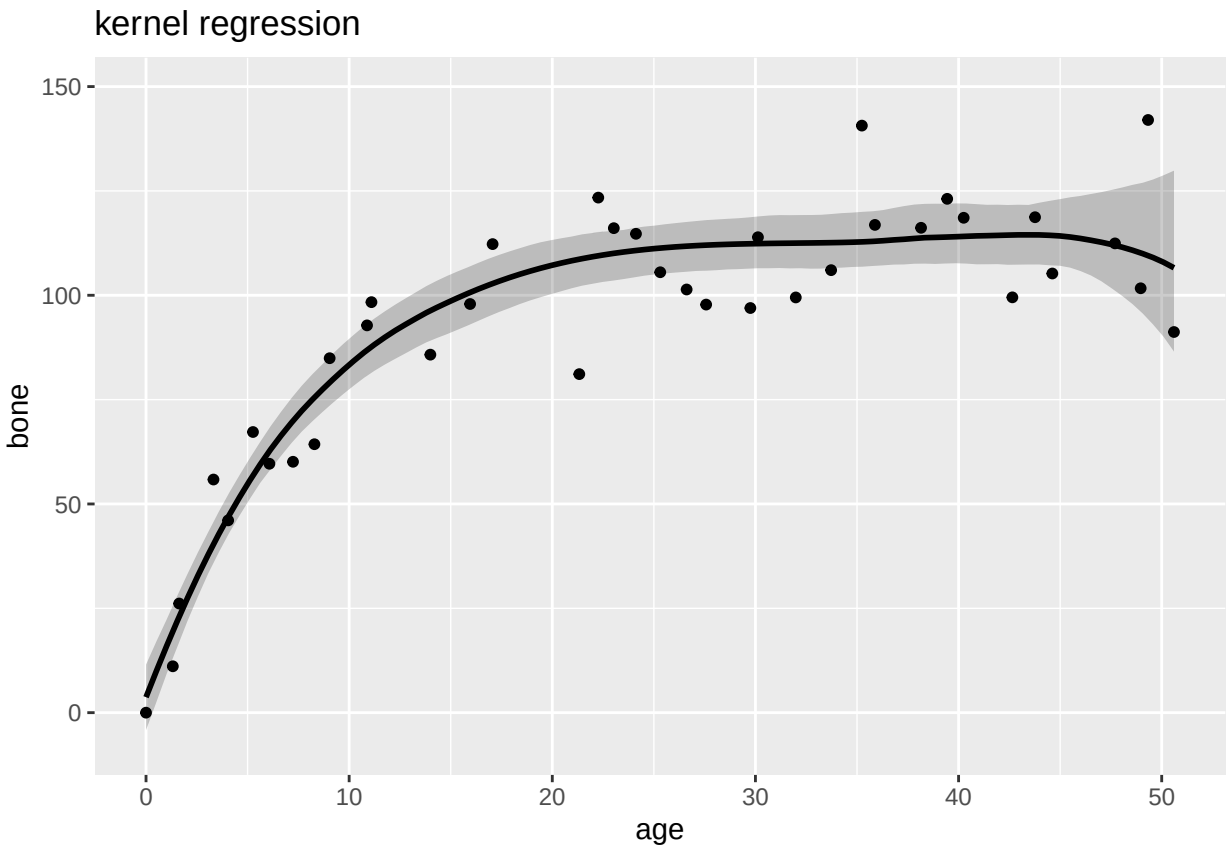


Comment: The optimum neighbour(2) was obtained through cross validation.The fitted curve followed the pattern of the data points but its wiggly. The MSE is 6.197896

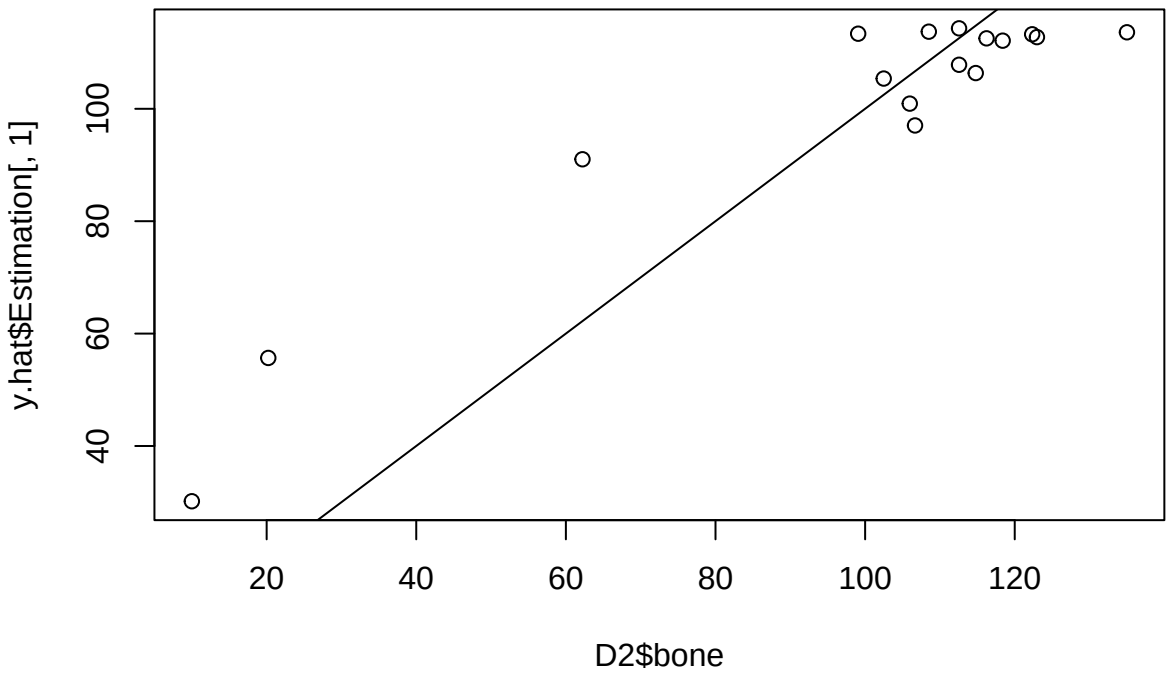
```
# (b) kernel regression
library("npregfast")
```

```
## Warning: package 'npregfast' was built under R version 4.2.2
```

```
fit.ker.reg <- frfast(bone ~ age, data = D1, seed = 130853,
    smooth = "kernel", kernel = "epanech", p = 3)
autoplot(fit.ker.reg, der = 0,pcol = "black",main = "kernel regression")
```




```
y.hat = predict(fit.ker.reg,newdata = D2)
plot(x=D2$bone,y.hat$Estimation[,1]);abline(a=0,b=1)
```




```
# plot(x=age,y=bone)
# lines(x=seq(0,50,length.out=16),y=y.hat$Estimation[,1],col="blue")
# lines(x=seq(0,50,length.out=16),y=y.hat$Estimation[,2],col="red")
# lines(x=seq(0,50,length.out=16),y=y.hat$Estimation[,3],col="red")


fit.ker.reg.MSE = mean((y.hat$Estimation[,1]-D2$bone)^2);fit.ker.reg.MSE
```

```
## [1] 227.7729
```

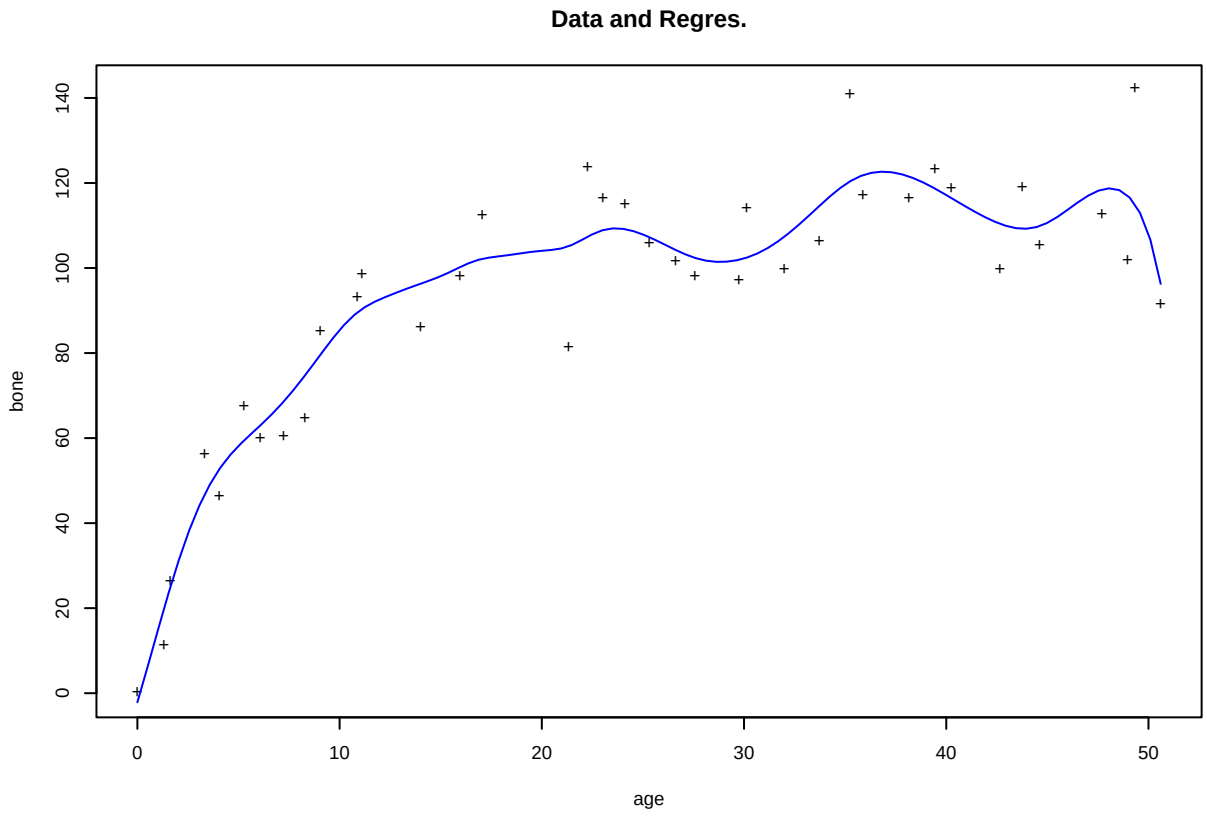
Comment: The Epanechnikov kernel was choose for the kernel regression,as it has a good approximation. Bandwidth was automatically picked by the the function frfast which is 0.72. The mse is 227.7729. Still some points are out of the confidence interval of the fitted curve.

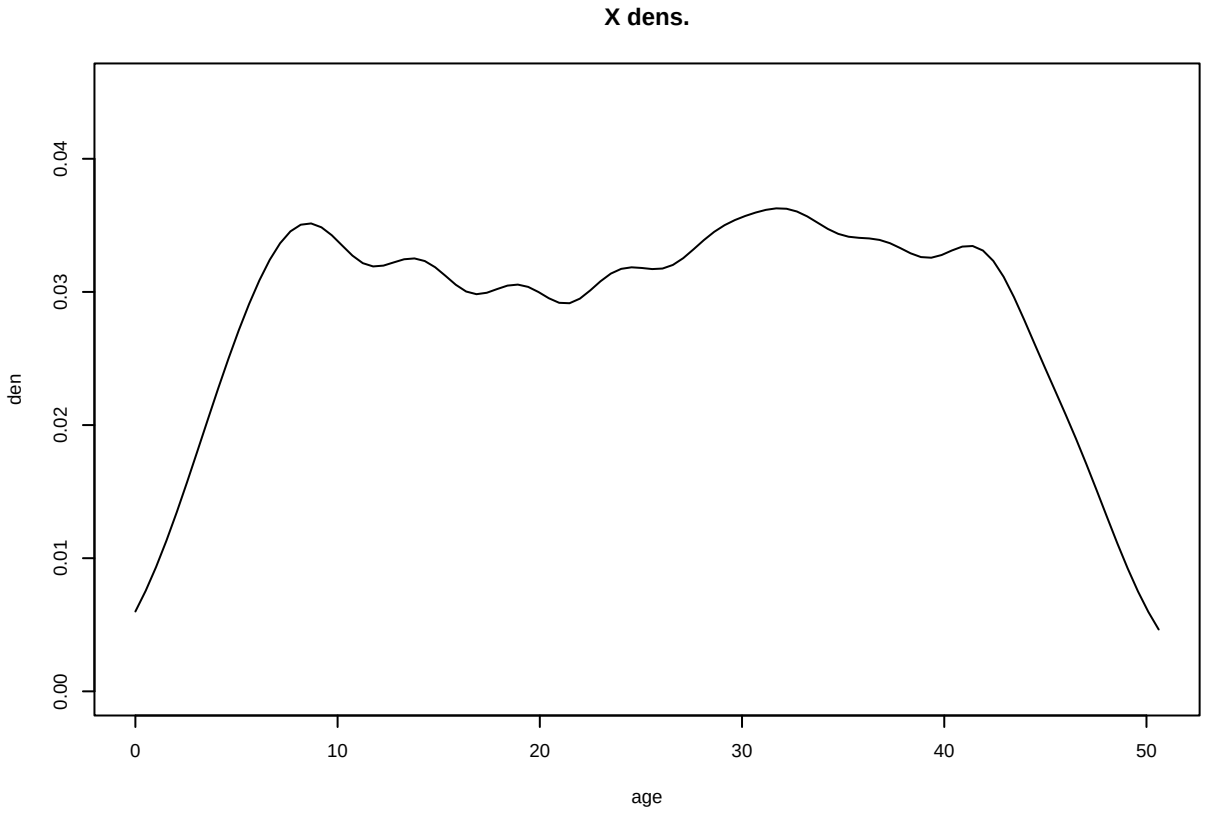
```
#c
#LOCAL POLYNOMIAL REGRESSION

 library(KernSmooth)
```

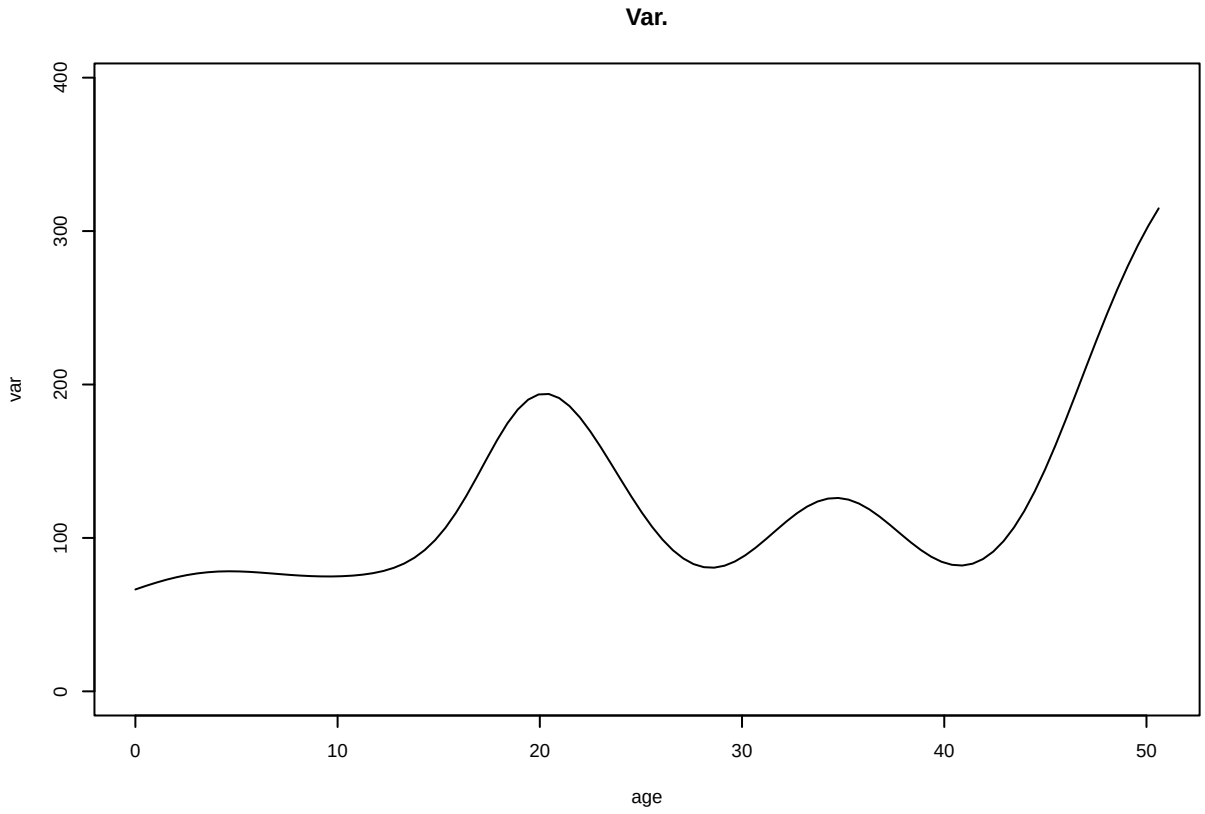
```
## Warning: package 'KernSmooth' was built under R version 4.2.2
```

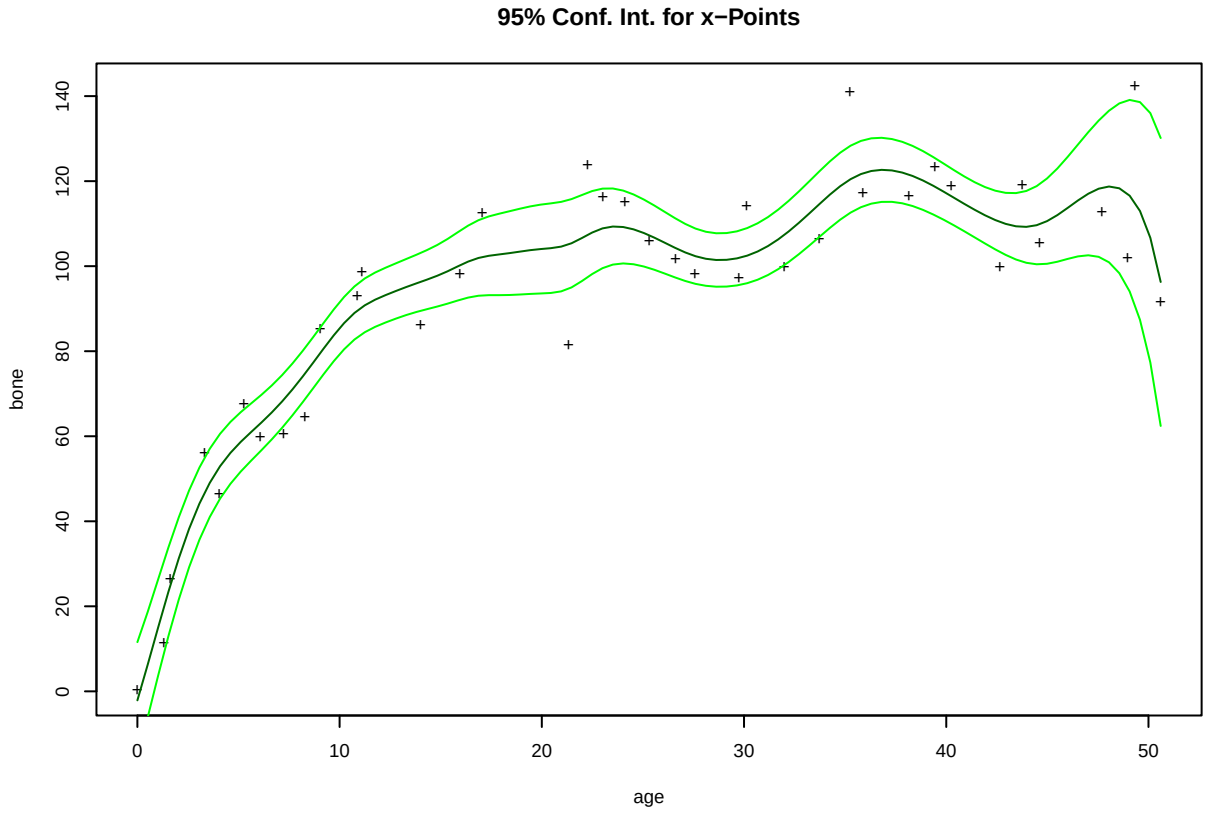
```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
# help(package="KernSmooth")
# plot(x=age, y=bone, xlab="age", ylab="bone",main = "local-GaussKernal reg")
 h <- dpill(age, bone)    # BANDWIDTH SELECTION (direct plug-in) FOR Gaussian kernel
# fit <- locpoly(age, bone, drv = 0L, degree=3,
#                kernel = "normal", bandwidth = h) # local linear Gaussian kernel regression
# lines(fit, col="red", lwd=2)

h <- dpill(age, bone) # BANDWIDTH SELECTION
library(locpol)
r <- locpol(bone~age, data=D1, deg=3, kernel=gaussK,bw=h)
plot(r,which = c(1,4))
```

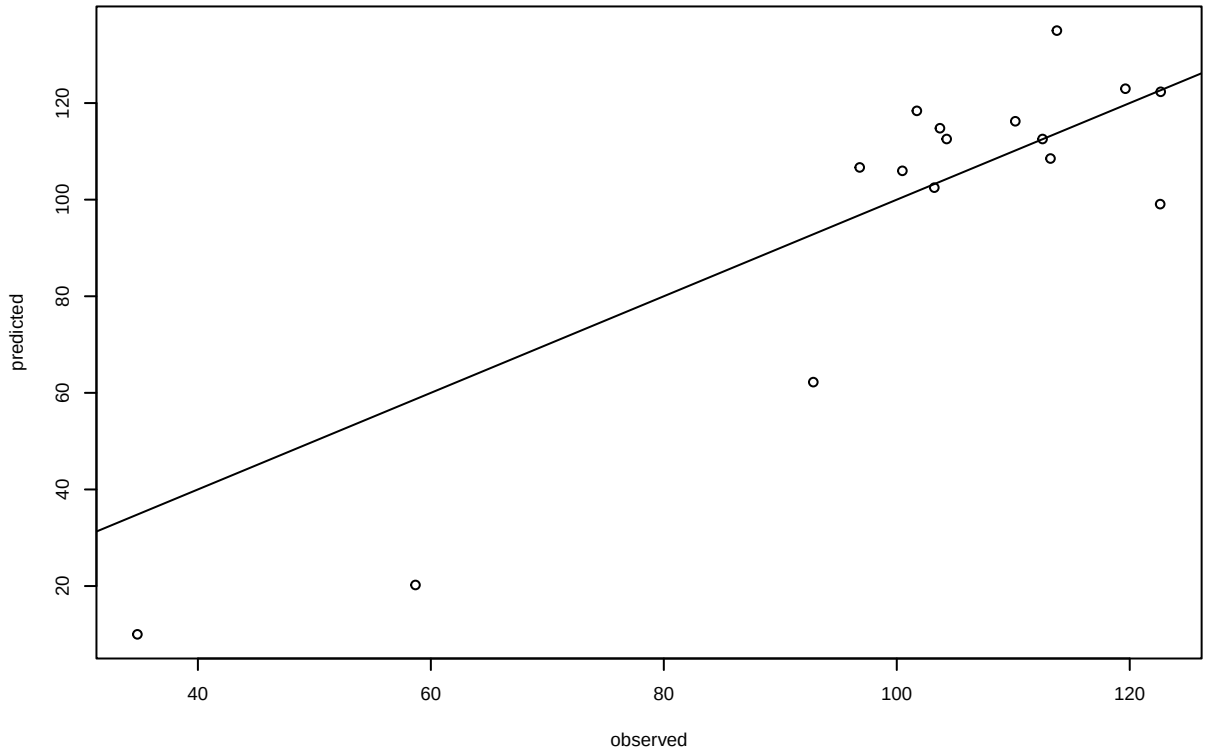



X dens.
den
0.00
0.01
0.02
0.03
0.04
0
10
20
30
40
50
age


Var.
var
age
0
100
200
300
400
0
10
20
30
40
50




```
r.pred = locpol(bone~age,data = D1,deg = 3,kernel = gaussK,bw=h,xeval = D2$age)
plot(y = D2$bone,x=fitted(r.pred),xlab="observed",ylab = "predicted");abline(a=0,b=1)
```




```
r.MSE=mean((D2$bone-fitted(r.pred))^2);r.MSE
```

```
## [1] 293.9948
```

```
#predict(fit,newdata=data.frame(age=D2$age))
```

Comment: Gaussian kernal was used. The bandwith was picked by the dpill function, which is 2.96. The MSE is 293.9948.

5 Regression/smoothing splines

Finally, regression/smoothing splines are applied.

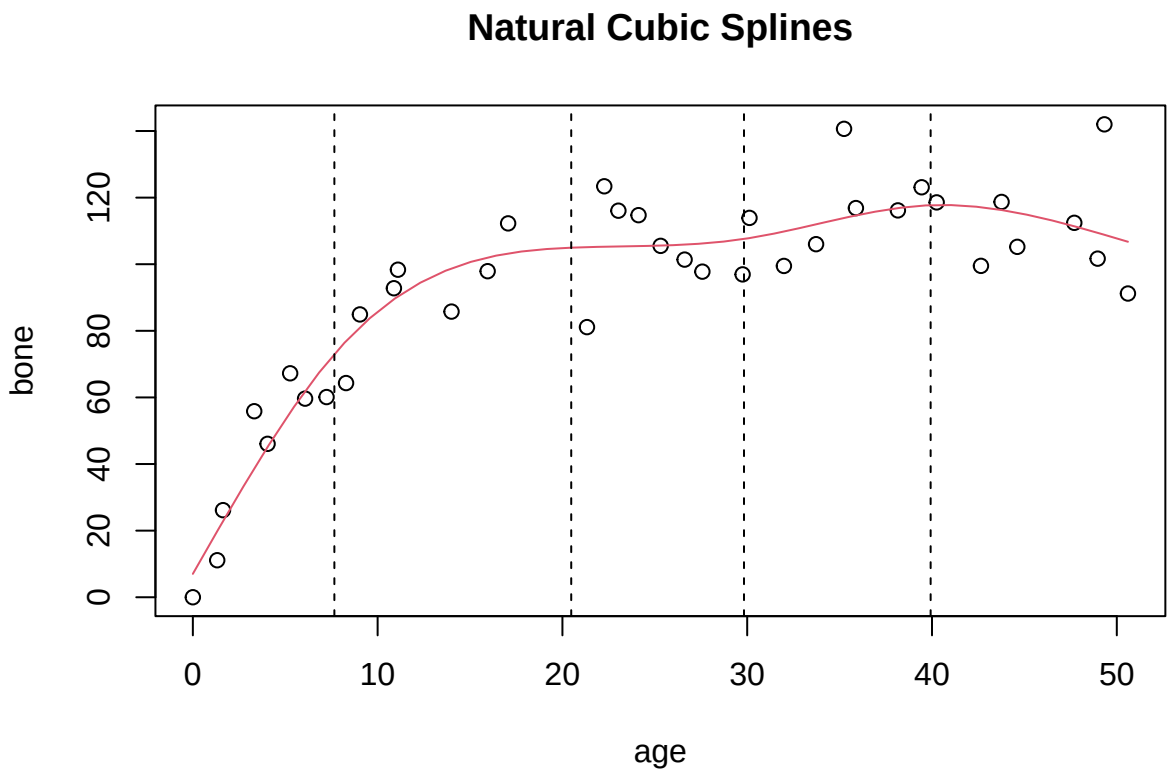
(a) Apply regression splines (e.g., natural cubic splines) to model the training data D1: Plot the resultant curve. Then use the fitted model to predict the test data D2. Plot the observed and predicted response values and obtain the prediction MSE on D2.

(b) Apply smoothing splines to D1: Always specify the choice of your kernel function and comment on how you determine the tuning parameter. Add the resultant curve to the scatterplot. Then apply the fitted model to the test data D2. Plot the observed and predicted response values and obtain the prediction MSE.

```r
# (a)
library("splines")
#ns(D1$age, df = 5)
summary(fm1 <- lm(bone ~ ns(age, df = 5), data = D1))
```

```
##
## Call:
## lm(formula = bone ~ ns(age, df = 5), data = D1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.023  -8.642   0.356   5.937  33.165
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.024      7.916   0.887    0.382
## ns(age, df = 5)1   99.603     12.577   7.919 4.88e-09 ***
## ns(age, df = 5)2   97.217     13.911   6.989 6.42e-08 ***
## ns(age, df = 5)3   91.239     11.108   8.214 2.21e-09 ***
## ns(age, df = 5)4  181.485     20.643   8.792 4.79e-10 ***
## ns(age, df = 5)5   49.587      8.736   5.676 2.78e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.82 on 32 degrees of freedom
## Multiple R-squared:  0.8659, Adjusted R-squared:  0.8449
## F-statistic: 41.32 on 5 and 32 DF,  p-value: 4.815e-13
```

```r
par(mfrow=c(1,1))
plot(bone~age, data=D1, xlab = "age", ylab = "bone",
     main="Natural Cubic Splines")
spd <- seq(min(bone), max(age), len = 38)
lines(spd, predict(fm1, data.frame(age=spd)), lty=1, col=2)
abline(v=c(7.658201, 20.476077, 29.825635, 39.926965),lty = 2)
```

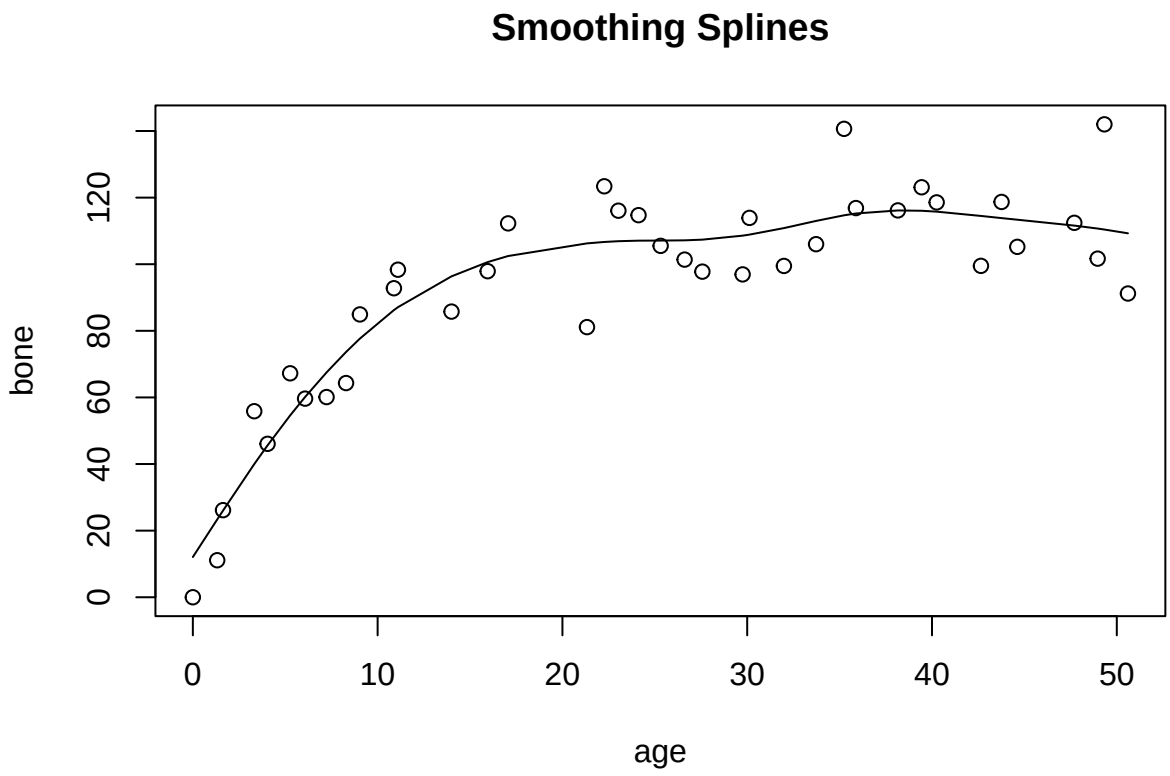



```
ns.pred = predict(fm1,data.frame(age=D2$age))
ns.MSE = mean((ns.pred-D2$bone)^2);ns.MSE
```

```
## [1] 240.3677
```

Comment: The prediction MSE is 240.3677.

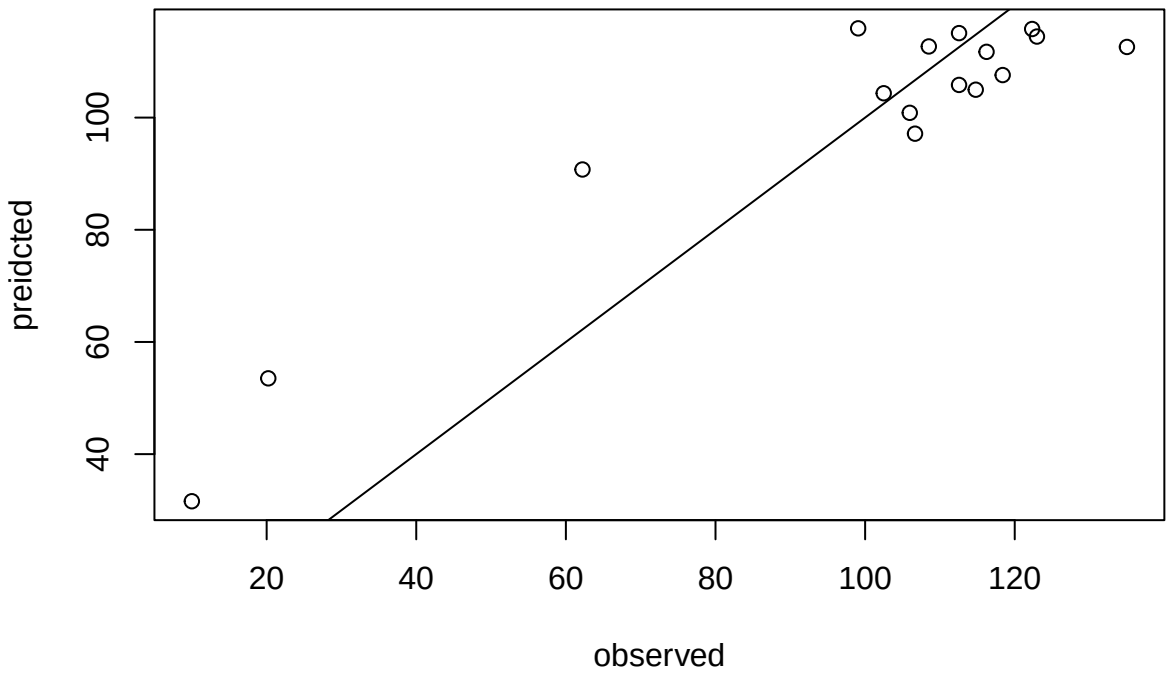
```
# (b)
fit.spl <- smooth.spline(age, bone, cv = FALSE); # Cv=F: GCV by default
plot(x=age, y = bone, main = "Smoothing Splines")
lines(fit.spl)
```




```
#df opt
fit.spl$df
```

```
## [1] 5.785626
```

```
spl.pred = predict(fit.spl,x=D2$age)$y
plot(x=D2$bone,y=spl.pred,xlab = "observed",ylab = "preidcted")
abline(a=0,b=1)
```




```
spl.MSE = mean((D2$bone - spl.pred )^2);spl.MSE
```

```
## [1] 232.0398
```

Comment: Natural cubic basis were used. The prediction MSE is 232.0398.

6 MSE measures

Tabulate all the prediction MSE measures. Which methods give favorable results?

```
# Table for all the MSEs.
Models = c("asym-exp -red","KNN","Kernal Reg","cubic local-reg","regsplines-NC","smoothing")

mse = c(jaw.mod.red.MSE, fit.knn.MSE,fit.ker.reg.MSE,r.MSE,ns.MSE,spl.MSE)

data.frame(Model=Models,MSE=round(mse,4))
```

```
##             Model      MSE
## 1   asym-exp -red 141.8087
## 2             KNN   6.1979
```

```
## 3      Kernal Reg 227.7729
## 4 cubic local-reg 293.9948
## 5   regsplines-NC 240.3677
## 6       smoothing 232.0398
```

Comment: The prediction MSE for KNN regression is the lowest among the models we constricted. For this model, predicted vs observed plot has points pretty close to y=x line. So, we can say this model works well for prediction among the others based on MSE criterion.