Project V (Parametric/Nonparametric Nonlinear Regression)

Quaye E, George

Due: 11/02/2020

Contents

1 Question 1- Bringing in Data

Bring in the data D and make a scatterplot of bone vs. age. Does their association look linear?

```
# Bring in the Data
Data <- read.table(file="jaws.txt", header = TRUE)
dim(Data)
```

```
## [1] 54  2
```

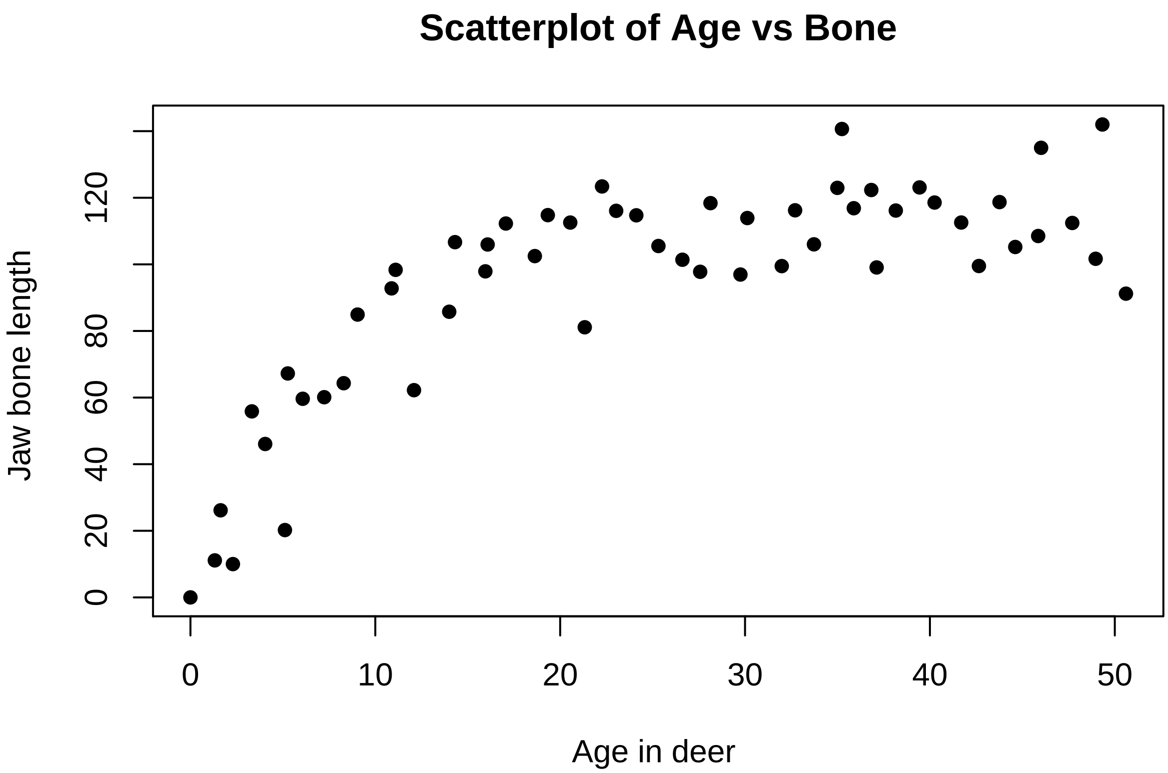
```
head(Data)
```

```
##          age      bone
## 1  0.000000   0.00000
## 2  5.112000  20.22000
## 3  1.320000  11.11130
## 4 35.240000 140.65000
## 5  1.632931  26.15218
## 6  2.297635  10.00100
```

The data has 52 observations with 2 columns.

Making a Scatter Plot of age vrs bone.

```
plot(Data$age, Data$bone, main="Scatterplot of Age vs Bone",
    xlab="Age in deer ", ylab="Jaw bone length ", pch=16)
```



Given the plot above, there appears a nonlinear relationship between the age of the deer and the jaw bone length.

2 Question 2- Partitioning

Randomly partition the data D into the training set D_1 and the test set D_2 with a ratio of approximately $2:1$ on the sample size.

```
#Partitioning data
set.seed(123)
sampleData <- sample(nrow(Data), (2.0/3.0)*nrow(Data), replace = FALSE) # training set
TrainData <- Data[sampleData, ]
#test set
TestData <- Data[-sampleData, ]
dim(TrainData)
```

```
## [1] 36  2
```

```
dim(TestData)
```

```
## [1] 18  2
```

The TrainData has 36 observations and the TestData has 18 both with 2 columns each.

3 Question 3- Parametric Nonlinear models

(a) Fit an asymptotic exponential model:

```
attach(TrainData)
model1 <- nls(bone ~ beta1 - beta2*exp(-beta3*age),
start=list(beta1 = 100, beta2 = 90, beta3 = 0.3), trace=T)
```

```
## 12019.56 :  100.0  90.0   0.3
## 7218.668 :  105.2541101  88.8367571   0.1157341
## 4897.436 :  114.2853948 109.4732635   0.1269187
## 4895.049 :  114.2167428 110.0838047   0.1267379
## 4895.049 :  114.2188418 110.0775461   0.1267163
## 4895.049 :  114.2190909 110.0767907   0.1267136
```

Given the above results, it is observed that the estimate for the parameters are $beta1 = 114.2190909$, $beta2 = 110.0767907$ and $beta3 = 0.1267136$ as the model converges

```
# Summary of the fitted model
summary(model1)
```

```
##
## Formula: bone ~ beta1 - beta2 * exp(-beta3 * age)
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 114.21909    3.29967  34.615  < 2e-16 ***
## beta2 110.07679   10.61256  10.372 6.43e-12 ***
## beta3   0.12671    0.02346   5.402 5.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.18 on 33 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 2.55e-06
```

```
detach(TrainData)
```

From the summary output and the P-values, all the coefficients are statistically significant at the level of $\alpha = 0.05$.

(b) Fit the reduced model

```r
# Fitting the reduced model
attach(TrainData)
model2 <- nls(bone ~ beta1*(1-exp(-beta3*age)),
  start=list(beta1 = 100, beta3 = 0.3), trace=T)
```

```
## 11009.56 :  100.0   0.3
## 6292.938 :  104.9752178   0.1705696
## 4972.119 :  113.0795051   0.1290611
## 4913.108 :  113.8081475   0.1333996
## 4913.027 :  113.7838427   0.1337274
## 4913.027 :  113.7811700   0.1337499
## 4913.027 :  113.7809830   0.1337514
```

```r
summary(model2)
```

```
##
## Formula: bone ~ beta1 * (1 - exp(-beta3 * age))
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 113.78098    2.97736  38.215  < 2e-16 ***
## beta3   0.13375    0.01507   8.874 2.26e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.02 on 34 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 1.151e-06
```

```r
detach(TrainData)
```

From the above results, we observe that the estimate for the parameters are $beta1 = 113.78098$ and $beta3 = 0.13375$ as the model converges at the 6th iteration. In the summary output, all the coefficients are statistically significant at the level of $\alpha = 0.05$ compared to the $p - values$.

Comparing the two models using anova function

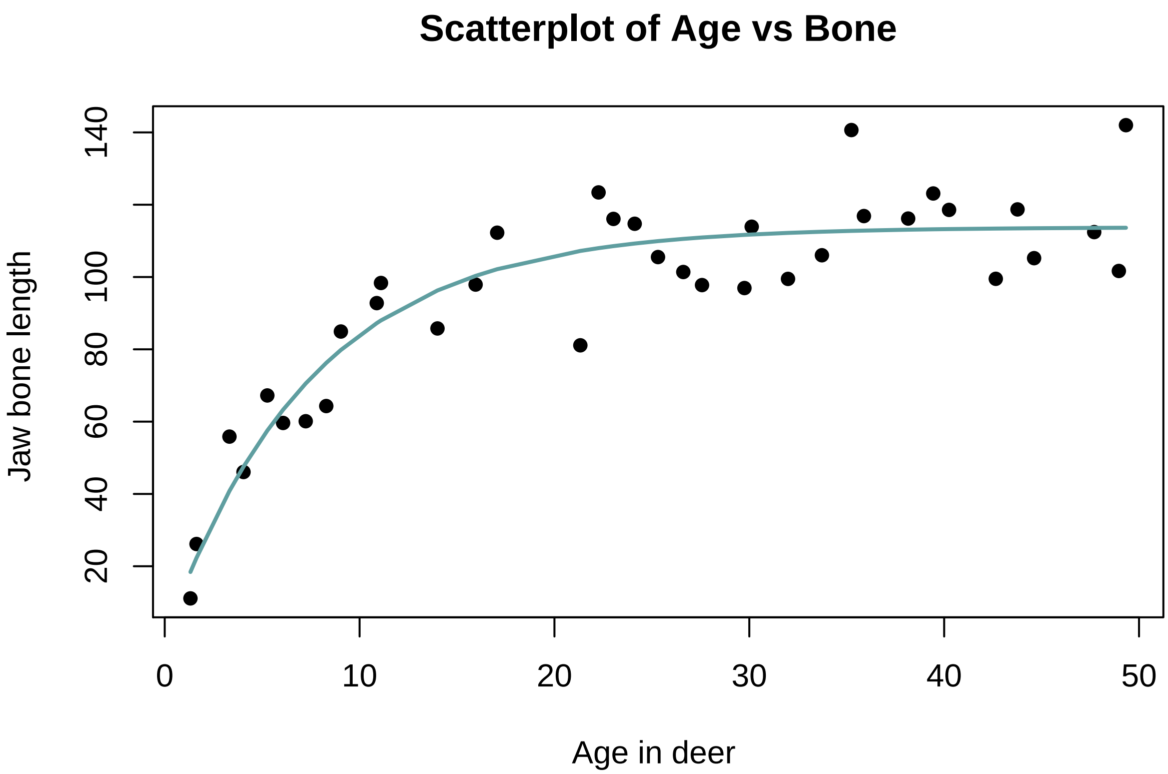
```r
# Compare the two models
anova(model2,model1)
```

```
## Analysis of Variance Table
##
## Model 1: bone ~ beta1 * (1 - exp(-beta3 * age))
## Model 2: bone ~ beta1 - beta2 * exp(-beta3 * age)
##   Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
## 1     34       4913
## 2     33       4895  1 17.978  0.1212   0.73
```

Given the above results, the $p-value$ of the test is 0.73. This indicates that the reduced model (model2) is not significantly (statistically) different from original model (model1) at the level of $\alpha = 0.05$. Hence a conclusion that the reduced model (model2) is better than model1 can be drawn.

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

```
# Fitted Curve vs Train Data
plot(TrainData$age, TrainData$bone, main="Scatterplot of Age vs Bone",
    xlab="Age in deer ", ylab="Jaw bone length ", pch=16)
lines(sort(TrainData$age), fitted.values(model2)[order(TrainData$age)], lwd=2,col="cade
```



The plot above is the line plot imposed on the scatterplot.

(d) Apply the better model in 2(b) to the test set D_2 and compute the prediction mean square error (MSE).

```r
# MEAN SQUARE ERROR FOR PREDICTION
Yhat.Pred.Ex <- predict(model2, newdata = TestData); Yhat.Pred.Ex
```

```
##  [1]   0.00000  56.35262  30.10403  91.20651  97.00133 100.52982 104.36376
##  [8] 105.20692 106.49329 111.13833 112.34819 112.72560 112.95545 112.98668
## [15] 113.35027 113.53412 113.53935 113.65018
```

```r
yobs <- TestData[, 2]
MSEP.ex <- mean((yobs-Yhat.Pred.Ex)^2)
MSEP.ex
```

```
## [1] 236.0823
```

Given the output, the prediction mean square error = 236.0823

4 Question 4- Local Regression Methods

(a) On basis of D_1; 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 D_2 and obtain the prediction MSE.

```
set.seed(123)
# Final optimal K via 10- fold CV
library("FNN")
SSEP <- function(yobs, yhat) sum((yobs-yhat)^2)

K <- 1:10
V <- 4
id.fold <- sample(1:V, size = NROW(TrainData), replace=T)
SSE <- rep(0, length(K))
for(k in 1:length(K)){
  for(v in 1:V){
    train1<- TrainData[id.fold!=v, ];
    train2<- TrainData[id.fold==v, ];
    yhat2 <- knn.reg(train=train1, y=train1$bone, test=train2, k=K[k], algorithm="kd_tre
    SSE[k] <- (SSE[k] + SSEP(train2$bone, yhat2))
  }
}
cbind(K, SSE)
```

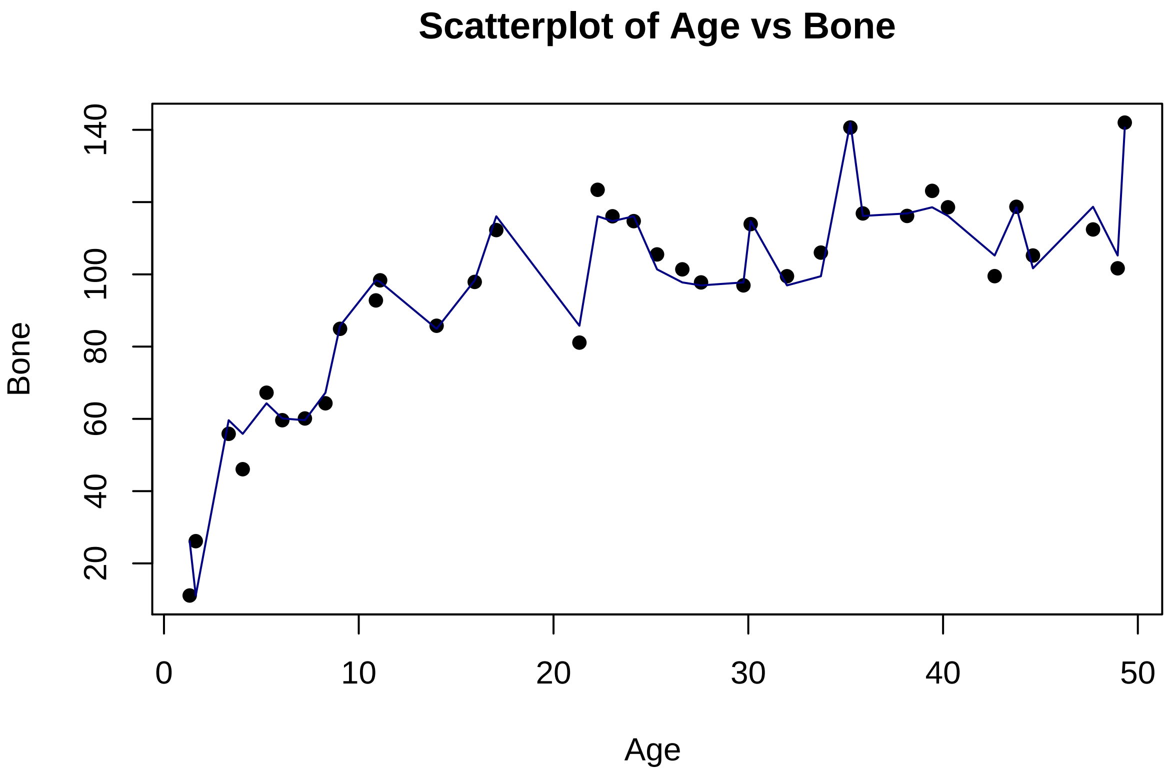
```
##        K      SSE
##  [1,]  1 1103.518
##  [2,]  2 1635.627
##  [3,]  3 2499.477
##  [4,]  4 3201.210
##  [5,]  5 4131.390
##  [6,]  6 4953.051
##  [7,]  7 6195.980
##  [8,]  8 7025.948
##  [9,]  9 8271.225
## [10,] 10 9165.624
```

Using the 10-fold CV, we see that K=1, gives the optimal K. Hence we choose the tuning parameter K=1.

```
# KNN regression
library("FNN")
fit.knn1 <- knn.reg(train=TrainData, y=TrainData$bone, k=1, algorithm="kd_tree");
```

```
plot(TrainData$age, TrainData$bone, main="Scatterplot of Age vs Bone",
    xlab="Age ", ylab="Bone", pch=16)

lines(sort(TrainData$age), fit.knn1$pred[order(TrainData$age)],lty=1, col="navyblue")
```



Shown in the above figure, the fitted curve from KNN is wiggly even for the optimal K = 1. This is because of its discontinuous weighting function.

```
# MEAN SQUARE ERROR FOR PREDICTION
fit.knn <- knn.reg(train=TrainData,test=TestData,y=TrainData$bone, k=1, algorithm="kd_tr
yobs <- TestData[, 2]
MSEP.knn <- mean((yobs-fit.knn$pred)^2)
MSEP.knn
```

```
## [1] 25.91247
```

Given the output above, the prediction mean square error = 25.91247.

(b) Apply kernel regression to obtain a nonlinear fit. State what your bandwidth is and how you decide on the choice. Obtain its prediction MSE on D_2.

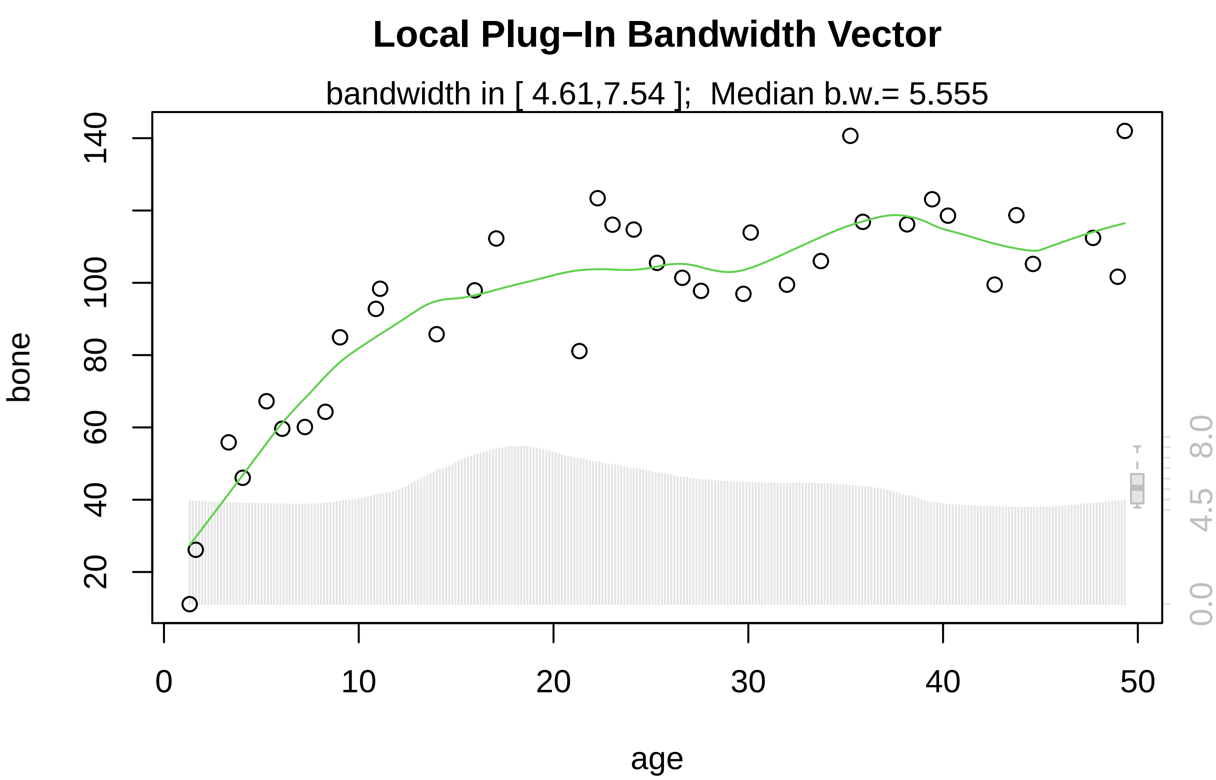
```r
# Kernel regression smoothing with adaptive local plug-in bandwidth selection.
library(lokern)

lofit <- lokerns(TrainData$age, TrainData$bone)
(sb <- summary(lofit$bandwidth))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.615   4.812   5.555   5.626   6.215   7.537
```

```r
op <- par(fg = "gray90", tcl = -0.2, mgp = c(3,.5,0))
plot(lofit$band, ylim=c(0,3*sb["Max."]), type="h", ann = F, axes = FALSE)
if(R.version$major > 1 || R.version$minor >= 3.0)
boxplot(lofit$bandwidth, add = TRUE, at = 304, boxwex = 8,
    col = "gray90", border="gray", pars = list(axes = FALSE))
axis(4, at = c(0,pretty(sb)), col.axis = "gray")
par(op); par(new=TRUE)

plot(bone ~ age, data = TrainData, main = "Local Plug-In Bandwidth Vector")
lines(lofit$x.out, lofit$est, col=3)
mtext(paste("bandwidth in [", paste(format(sb[c(1,6)], dig = 3),collapse=","),
    "];  Median b.w.=",formatC(sb["Median"])))
```



The bandwidth h is the scaling factor that controls how wide the probability mass is spread around a point and affects the smoothness or roughness of the resultant estimate. The bandwidth is given by the local bandwidth array for kernel regression estimation. In this case, the bandwidth is within the interval $[4.61, 7.54]$.

```
# MEAN SQUARE ERROR FOR PREDICTION
Yhat.Pred.kernel <- predict(lofit, newdata = TestData);
```

```
## using first column of data.frame as 'x'
```

```
yobs <- TestData[, 2]
MSEP.kernel <- mean((yobs-Yhat.Pred.kernel$y)^2)
MSEP.kernel
```

```
## [1] 1619.398
```

From the output, the prediction mean square error = 1619.398

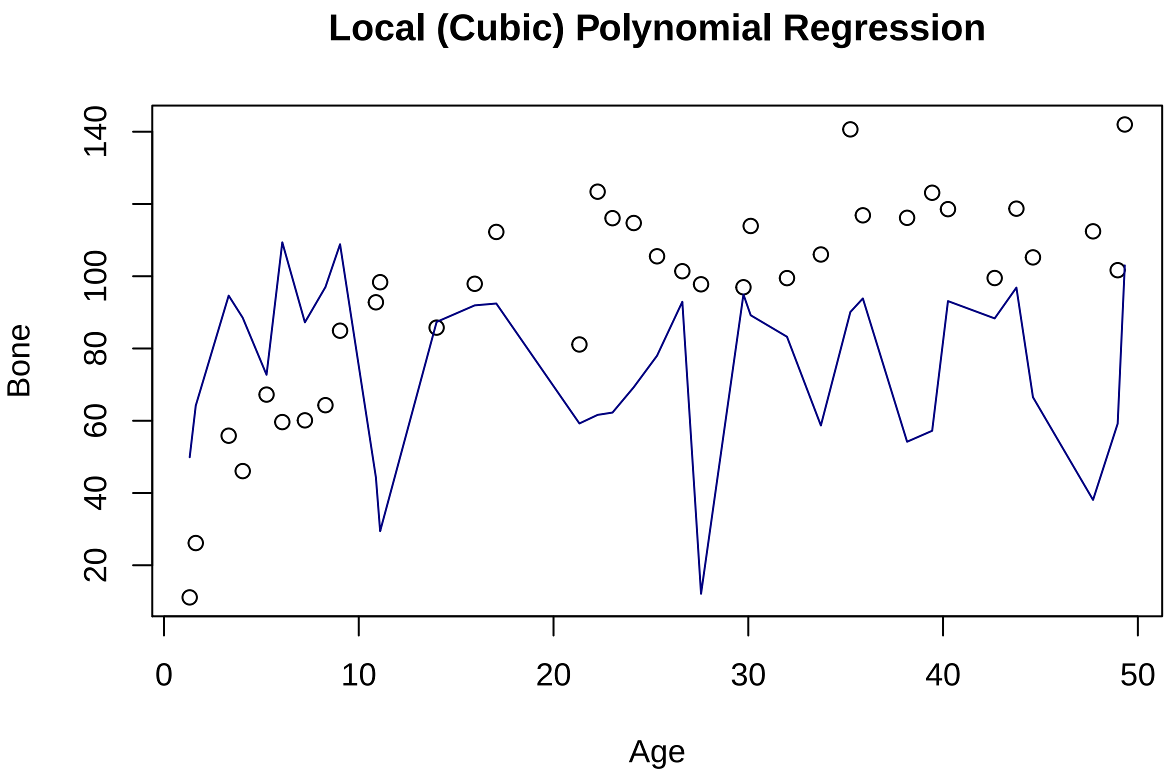
(c) Apply local (cubic) polynomial regression to the data. Plot and obtain its prediction MSE on D_2.

```
# Local (Cubic) Polynomial Regression
library(locpol)
fit.local <- locpol(bone~age, data=TrainData, deg=3, kernel=EpaK,bw =5)

Yhat.Pred.local <- locpol(bone~age, data=TrainData, xeval=TestData$age, deg=3, kernel=Ep
```

The Smoothing parameter, bandwidth chosen is bw = 5.

```
plot(TrainData$age, TrainData$bone, xlab = "Age", ylab = "Bone", main="Local (Cubic) Pol
lines(sort(TrainData$age), fitted(fit.local)[order(TrainData$age)], lty=1, col="navyblue
```



Given above is the plot of Local (Cubic) Polynomial Regression.

```
#MEAN SQUARE ERROR FOR PREDICTION
yobs <- TestData[, 2]
MSEP.local <- mean((yobs-Yhat.Pred.local)^2)
MSEP.local
```

```
## [1] 12482.95
```

From the output, the prediction mean square error = 12482.95

5 Question 5- Regression/Smoothing Splines

(a) Apply regression splines (e.g., natural cubic splines) to model the data. Plot the resultant curve and obtain its prediction MSE on D_2.

```
# Natural Cubic Splines
library(splines)
attach(TrainData)
bs(TrainData$age, df = 5)
```

```
##                        1            2            3            4            5
```

```
##  [1,] 7.382017e-03 3.548663e-01 5.452988e-01 9.245288e-02 0.0000000000
##  [2,] 5.377846e-01 3.904632e-01 4.458014e-02 0.000000e+00 0.0000000000
##  [3,] 0.000000e+00 1.579701e-04 1.313027e-02 2.508903e-01 0.7358214373
##  [4,] 5.477804e-01 3.789348e-01 4.166835e-02 0.000000e+00 0.0000000000
##  [5,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.0000000000
##  [6,] 0.000000e+00 5.097512e-02 3.880391e-01 5.239618e-01 0.0370240001
##  [7,] 0.000000e+00 3.532401e-02 3.280985e-01 5.676470e-01 0.0689305058
##  [8,] 0.000000e+00 1.387108e-01 5.487334e-01 3.122243e-01 0.0003315217
##  [9,] 0.000000e+00 1.778740e-06 6.921065e-04 6.325174e-02 0.9360543768
## [10,] 8.417982e-02 5.605343e-01 3.443330e-01 1.095278e-02 0.0000000000
## [11,] 6.486711e-02 5.381942e-01 3.800569e-01 1.688171e-02 0.0000000000
## [12,] 5.147309e-02 5.169309e-01 4.085211e-01 2.307499e-02 0.0000000000
## [13,] 6.498477e-02 6.636947e-04 1.460960e-06 0.000000e+00 0.0000000000
## [14,] 0.000000e+00 3.824849e-03 9.669041e-02 5.281041e-01 0.3713806236
## [15,] 3.576413e-02 4.826907e-01 4.472606e-01 3.428456e-02 0.0000000000
## [16,] 5.378243e-01 8.943171e-02 2.929337e-03 0.000000e+00 0.0000000000
## [17,] 2.262313e-02 4.409417e-01 4.862202e-01 5.021499e-02 0.0000000000
## [18,] 1.158088e-05 1.904601e-01 5.771989e-01 2.323295e-01 0.0000000000
## [19,] 4.318005e-01 4.519192e-02 9.634184e-04 0.000000e+00 0.0000000000
## [20,] 0.000000e+00 2.731611e-02 2.893128e-01 5.870020e-01 0.0963691368
## [21,] 3.453717e-01 2.524049e-02 3.826181e-04 0.000000e+00 0.0000000000
## [22,] 0.000000e+00 8.879902e-02 4.826749e-01 4.207875e-01 0.0077385632
## [23,] 1.251881e-02 3.922210e-01 5.228725e-01 7.238768e-02 0.0000000000
## [24,] 2.719707e-01 5.786397e-01 1.493758e-01 1.387101e-05 0.0000000000
## [25,] 0.000000e+00 1.021370e-01 5.053181e-01 3.884879e-01 0.0040570204
## [26,] 0.000000e+00 1.088703e-02 1.772930e-01 5.942188e-01 0.2176012251
## [27,] 3.783434e-01 5.236908e-01 9.715439e-02 0.000000e+00 0.0000000000
## [28,] 6.167471e-01 1.814084e-01 9.872999e-03 0.000000e+00 0.0000000000
## [29,] 8.652195e-04 2.562377e-01 5.800321e-01 1.628650e-01 0.0000000000
## [30,] 2.210875e-01 5.935694e-01 1.850701e-01 2.729028e-04 0.0000000000
## [31,] 6.205017e-01 2.372635e-01 1.615022e-02 0.000000e+00 0.0000000000
## [32,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 1.0000000000
## [33,] 5.828214e-01 1.246564e-01 5.125782e-03 0.000000e+00 0.0000000000
## [34,] 6.099239e-01 2.783313e-01 2.194289e-02 0.000000e+00 0.0000000000
## [35,] 0.000000e+00 6.293161e-03 1.296806e-01 5.659141e-01 0.2981120848
## [36,] 1.323685e-03 2.702513e-01 5.775189e-01 1.509062e-01 0.0000000000
## attr(,"degree")
## [1] 3
## attr(,"knots")
## 33.33333% 66.66667%
##  15.30234  32.56644
## attr(,"Boundary.knots")
## [1]  1.32000 49.32956
## attr(,"intercept")
## [1] FALSE
```

```
## attr(,"class")
## [1] "bs"     "basis"  "matrix"
```

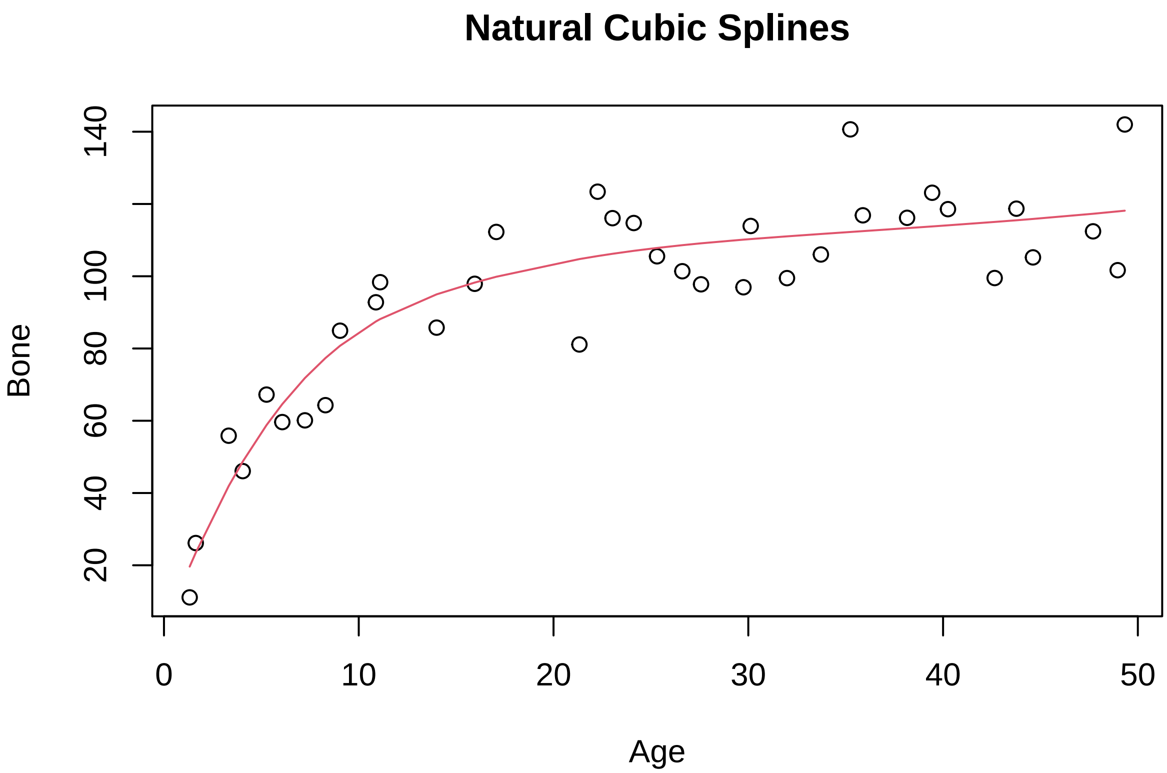
```
fm1 <- lm(bone ~ bs(age, df = 5), degree=3, data = TrainData)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'degree' will be disregarded
```

```
summary(fm1)
```

```
##
## Call:
## lm(formula = bone ~ bs(age, df = 5), data = TrainData, degree = 3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.643  -9.567   1.170   7.885  28.400
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)         19.63       8.48   2.315  0.02764 *
## bs(age, df = 5)1    58.23      16.83   3.460  0.00164 **
## bs(age, df = 5)2    84.76      15.24   5.563 4.76e-06 ***
## bs(age, df = 5)3    91.96      18.15   5.068 1.92e-05 ***
## bs(age, df = 5)4    95.48      14.45   6.610 2.57e-07 ***
## bs(age, df = 5)5    98.51      12.10   8.142 4.34e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.66 on 30 degrees of freedom
## Multiple R-squared:  0.8423, Adjusted R-squared:  0.816
## F-statistic: 32.05 on 5 and 30 DF,  p-value: 3.593e-11
```

```
par(mfrow=c(1,1))
plot(bone ~ age, data = TrainData, xlab = "Age", ylab = "Bone", main = "Natural Cubic Sp
spd <- seq(min(TrainData$age), max(TrainData$age), len = 36)
lines(sort(TrainData$age), fm1$fitted.values[order(TrainData$age)], lty=1, col=2)
```




```
detach()
```

In this case, B-Spline chooses 5 knots at suitable quantiles of age.

```
# MEAN SQUARE ERROR FOR PREDICTION
Yhat.Pred.spline <- predict(fm1,TestData)
```

```
## Warning in bs(age, degree = 3L, knots = c(`33.33333%` = 15.30233913, `66.66667%`
## = 32.56643584: some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
yobs <- TestData[, 2]
MSEP.spline <- mean((yobs-Yhat.Pred.spline)^2)
MSEP.spline
```

```
## [1] 263.8793
```

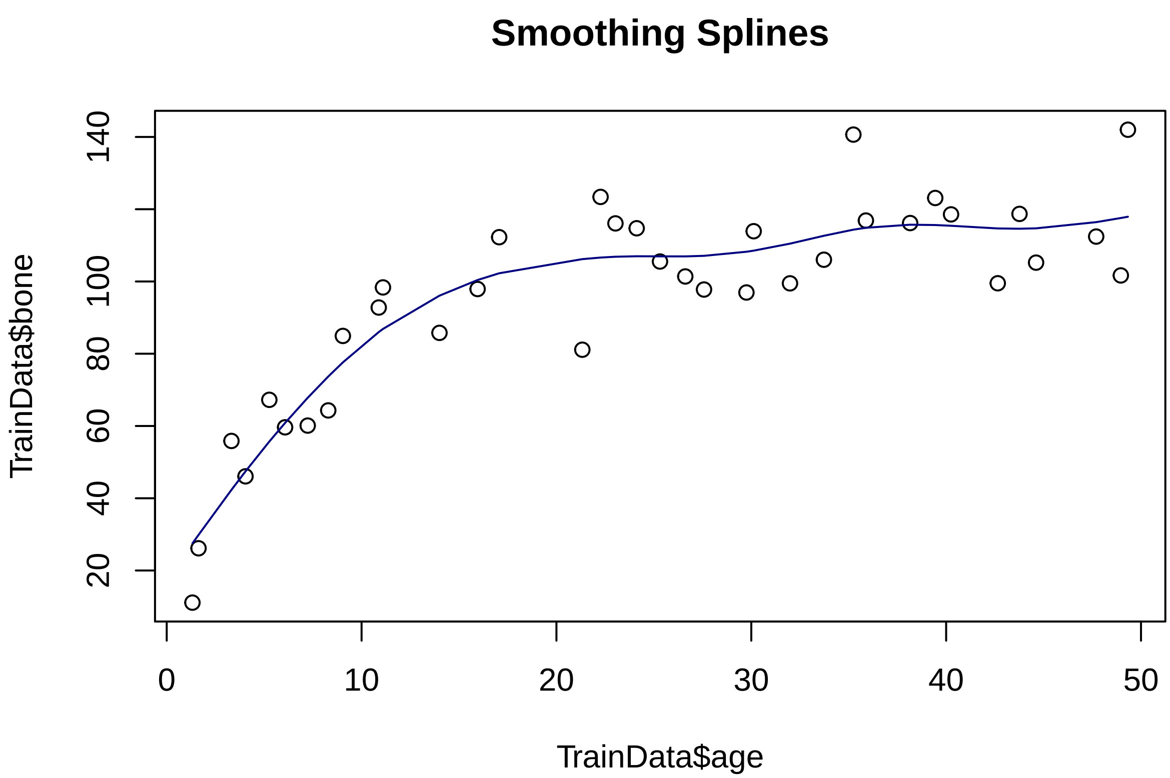
From the output, the prediction mean square error = 263.8793.

(b) Apply smoothing splines. Comment on how you determine the tuning parameter. Plot the resultant curve and obtain its prediction MSE on D_2.

```r
# Smoothing Splines
plot(TrainData$age, TrainData$bone, main = "Smoothing Splines")
fitopt <- smooth.spline(TrainData$age, TrainData$bone);fitopt
```

```
## Call:
## smooth.spline(x = TrainData$age, y = TrainData$bone)
##
## Smoothing Parameter  spar= 0.7000055  lambda= 0.001192558 (14 iterations)
## Equivalent Degrees of Freedom (Df): 5.750337
## Penalized Criterion (RSS): 4637.16
## GCV: 2154.222
```

```r
lines(fitopt, col = "navyblue")
```



Given this scenario, the generalized cross-validation (GCV) was used for the smoothing parameter estimation and a tuning parameter df = 5.750337 was used.

```r
# MEAN SQUARE ERROR FOR PREDICTION
Yhat.Pred.smooth <- predict(fitopt,TestData$age)
yobs <- TestData[, 2]
MSEP.smooth <- mean((yobs-Yhat.Pred.smooth$y)^2)
MSEP.smooth
```

```
## [1] 275.0591
```

Given the output, the prediction mean square error = 275.0591.

6 Question 6- Prediction MSE Results

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

```
Measure <- c(MSEP.ex,MSEP.knn,MSEP.kernel,MSEP.local,MSEP.spline,MSEP.smooth)
Measures <- data.frame("Method"= c("Asymptotic exponential model","KNN regression","Kern
knitr::kable(Measures, align = "lc")
```

Method	Prediction.MSE.Measures
Asymptotic exponential model	236.08228
KNN regression	25.91247
Kernel regression	1619.39755
Local cubic polynomial	12482.95403
Natural cubic spline	263.87930
Smoothing Splines	275.05908

The KNN regression method gives the favorable results from the output above.