Project V: Parametric/Nonparametric Nonlinear Regression

Appiah Prince* University of Texas at El Paso (UTEP)

November 08, 2022

Contents

*pappiah@miners.utep.edu

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 Bring in the data and make a scatterplot of bone vs. age.

```
jaw_data <- read.table(file="jaws.txt", header = TRUE)
dim(jaw_data)
```

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

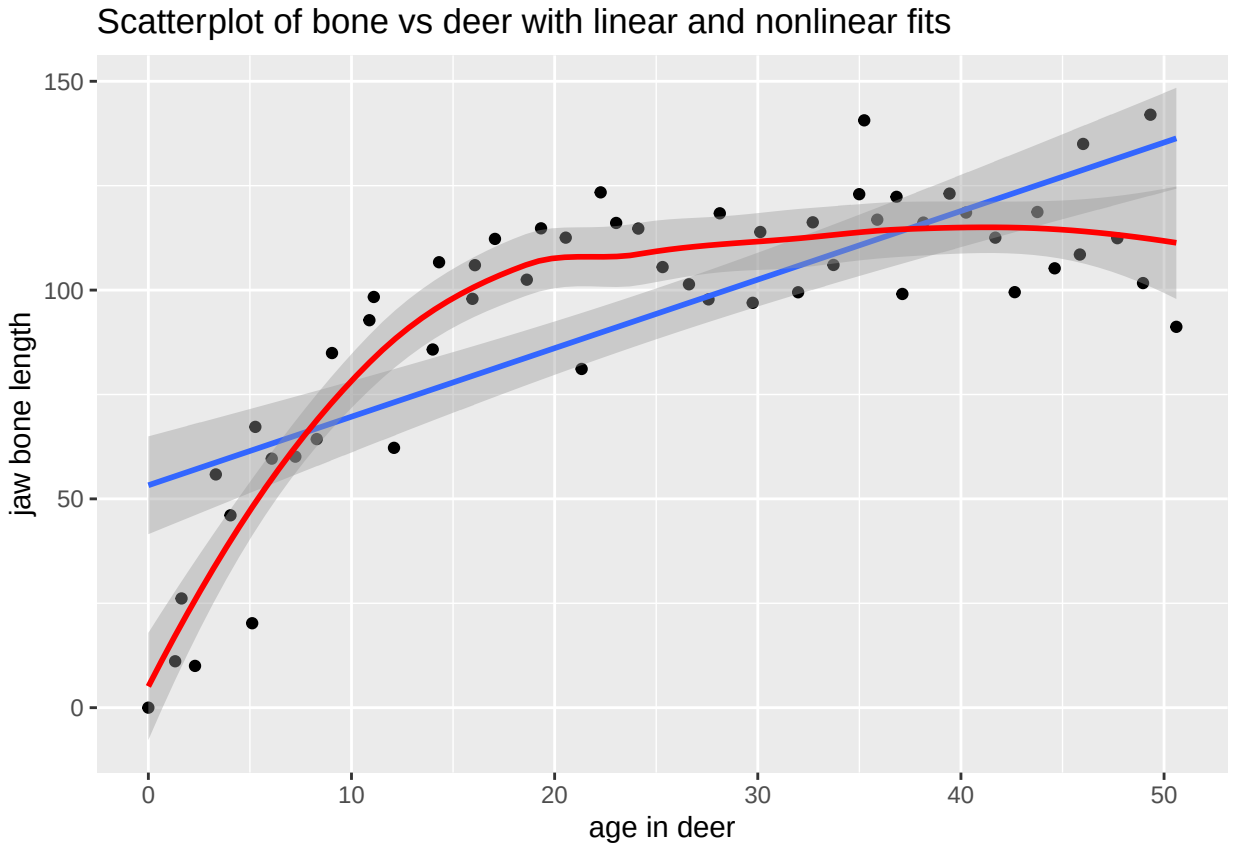
```
head(jaw_data)
```

```
##         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
```

Scatterplot of bone vs deer

```
library(ggplot2)
ggplot(data = jaw_data, mapping = aes(x = age, y = bone)) +
  geom_point()+
  geom_smooth(method = lm) +
  geom_smooth(method = "loess", col = "red") +
  ggtitle("Scatterplot of bone vs deer with linear and nonlinear fits") +
  xlab("age in deer") + ylab("jaw bone length")
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



Comment

- Almost all the points do not lie on both the linear and non linear fits. Most of the points are also not near the curve and the line and they do not fit the data very well.
- The associations of both the linear and non linear fits do not look linear.

2 Data Partitioning

2.1 Partition the data into train and test in the ratio 2:1 respectively

```
set.seed(126)
sample_jawdata <- sample(nrow(jaw_data), (2.0/3.0)*nrow(jaw_data), replace = FALSE)
train_jawdata <- jaw_data[sample_jawdata, ] # training set
test_jawdata <- jaw_data[-sample_jawdata, ] #test set
dim(train_jawdata)
```

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

```
dim(test_jawdata)
```

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

Comment

- The training set has dimension 36 rows(observations) and 2 columns(variables).
- The test set has dimension 18 rows(observations) and 2 columns(variables).

2.2 Checking for range of age of the train and test data to prevent extrapolation

```
range_age_original <- range(jaw_data$age)
range_age_train <- range(train_jawdata$age)
range_age_test <- range(test_jawdata$age)

range_age <- data.frame('original data'=range_age_original,
                        'train data' =range_age_train,
                        'test data'=range_age_test)
row.names(range_age) <- c("min","max")
range_age
```

```
##     original.data train.data test.data
## min        0.0000     0.0000   5.11200
## max       50.6041    50.6041  47.70016
```

Comment

- We see that the minimum and maximum age in original data are the same as the minimum and maximum age in the training set.
- We also observe that the range of age in the test set does not exceed that in the training set.
- Hence,the problem of extrapolation when it comes to prediction is prevented.

3 Parametric Nonlinear Models

3.1 Fitting an asymptotic exponential model

```
bone_jaw_model <- nls(bone ~ beta1 - beta2*exp(-beta3*age),data=train_jawdata,
    start=list(beta1 =120, beta2 =5, beta3 = 0.3), trace=T)
```

```
## 68418.84    (3.11e+00): par = (120 5 0.3)
## 62669.30    (3.10e+00): par = (119.7646 8.194851 0.1036513)
## 56133.24    (2.92e+00): par = (119.5195 15.04651 0.1227629)
## 44506.30    (2.56e+00): par = (119.1191 28.03725 0.1277773)
## 27630.39    (1.92e+00): par = (118.4202 50.78591 0.1280919)
## 11336.23    (9.59e-01): par = (117.3715 84.90946 0.1280788)
## 5905.352    (8.58e-06): par = (116.3228 119.033 0.1280811)
```

```
summary(bone_jaw_model)
```

```
##
## Formula: bone ~ beta1 - beta2 * exp(-beta3 * age)
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 116.32280    3.52183  33.029  < 2e-16 ***
## beta2 119.03297    8.50519  13.995 1.97e-15 ***
## beta3   0.12808    0.02239   5.721 2.19e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.38 on 33 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 8.58e-06
```

Comment

All the coefficients are statistically significant according to their p-values at 5% significant level. Also, the number of iterations to convergence is 6 with the Achieved convergence tolerance of 8.58e-06.

3.2 Fitting the reduced model under H0.

```
reduced_bone_jaw_model <- nls(bone ~ beta1*(1-exp(-beta3*age)),
            data=train_jawdata,start=list(beta1 = 120, beta3 = 0.3), trace=T)
```

```
## 17590.24    (1.31e+00): par = (120 0.3)
## 7668.779    (5.43e-01): par = (116.1738 0.1773253)
## 6155.665    (1.98e-01): par = (115.5184 0.1138522)
## 5923.009    (8.58e-03): par = (116.5345 0.1234269)
## 5922.574    (8.39e-06): par = (116.6298 0.1236428)
```

```
summary(reduced_bone_jaw_model)
```

```
##
## Formula: bone ~ beta1 * (1 - exp(-beta3 * age))
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## beta1 116.62976    3.42238   34.08  < 2e-16 ***
## beta3   0.12364    0.01675    7.38 1.48e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.2 on 34 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 8.394e-06
```

Comment

All the coefficients are statistically significant according to their p-values at 5% significant level. Also, the number of iterations to convergence is 4 with the Achieved convergence tolerance of 8.394e-06.

```
anova(bone_jaw_model, reduced_bone_jaw_model)
```

```
## 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     33     5905.4
## 2     34     5922.6 -1 -17.222  0.0962 0.7583
```

Comment

The p-value 0.7583 is greater than 0.05 which implies that we fail to reject H0.

```
AIC1 <- AIC(bone_jaw_model)
BIC1 <- BIC(bone_jaw_model)
AIC2 <- AIC(reduced_bone_jaw_model)
BIC2 <- BIC(reduced_bone_jaw_model)

AICS <- c(AIC1,AIC2,BIC1,BIC2)
compare <- data.frame(Methods=c('AIC non-reduced model','AIC reduced model',
                                'BIC non-reduced model','BIC reduced model'),
                                'Values' =AICS )
knitr::kable(compare, align = "lc", caption = "Comparing the two nls models
             with AIC and BIC ")
```

Table 1: Comparing the two nls models with AIC and BIC

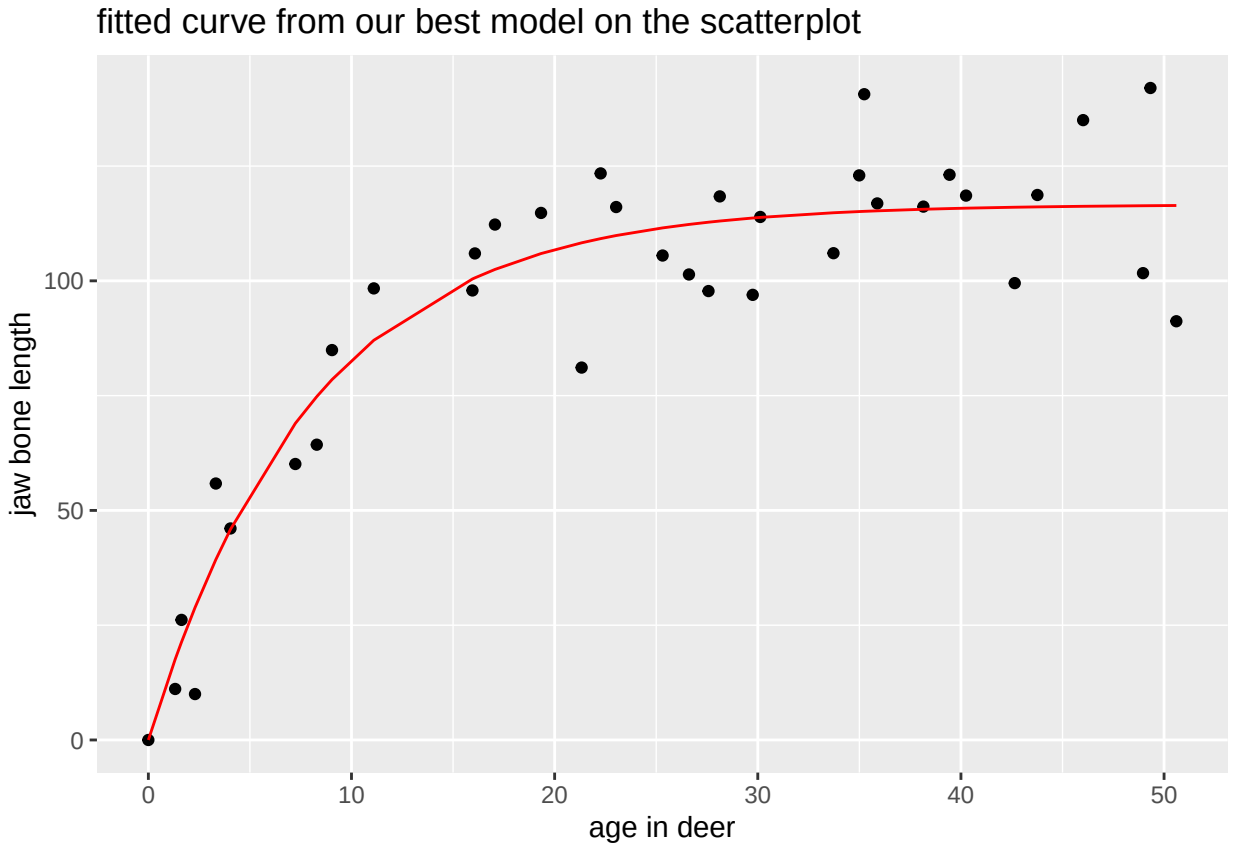
Methods	Values
AIC non-reduced model	293.7670
AIC reduced model	291.8718
BIC non-reduced model	300.1011
BIC reduced model	296.6224

Comment

Both the AIC and BIC of the reduced model are smaller than that of the non-reduced models'. Thus, this result together with the result from the anova function and the reduced model's parsimony confirm that the reduced model is better that the non-reduced model.

3.3 Based on the better model, add the fitted curve to the scatterplot.

```
fitted_values <- fitted.values(reduced_bone_jaw_model)
library(ggplot2)
ggplot(data = train_jawdata, mapping = aes(x = age, y = bone)) +
  geom_point()+
  geom_line(aes(x=age, y=fitted_values), col = 'red') +
  ggtitle("fitted curve from our best model on the scatterplot") +
  xlab("age in deer") + ylab("jaw bone length")
```



Comment

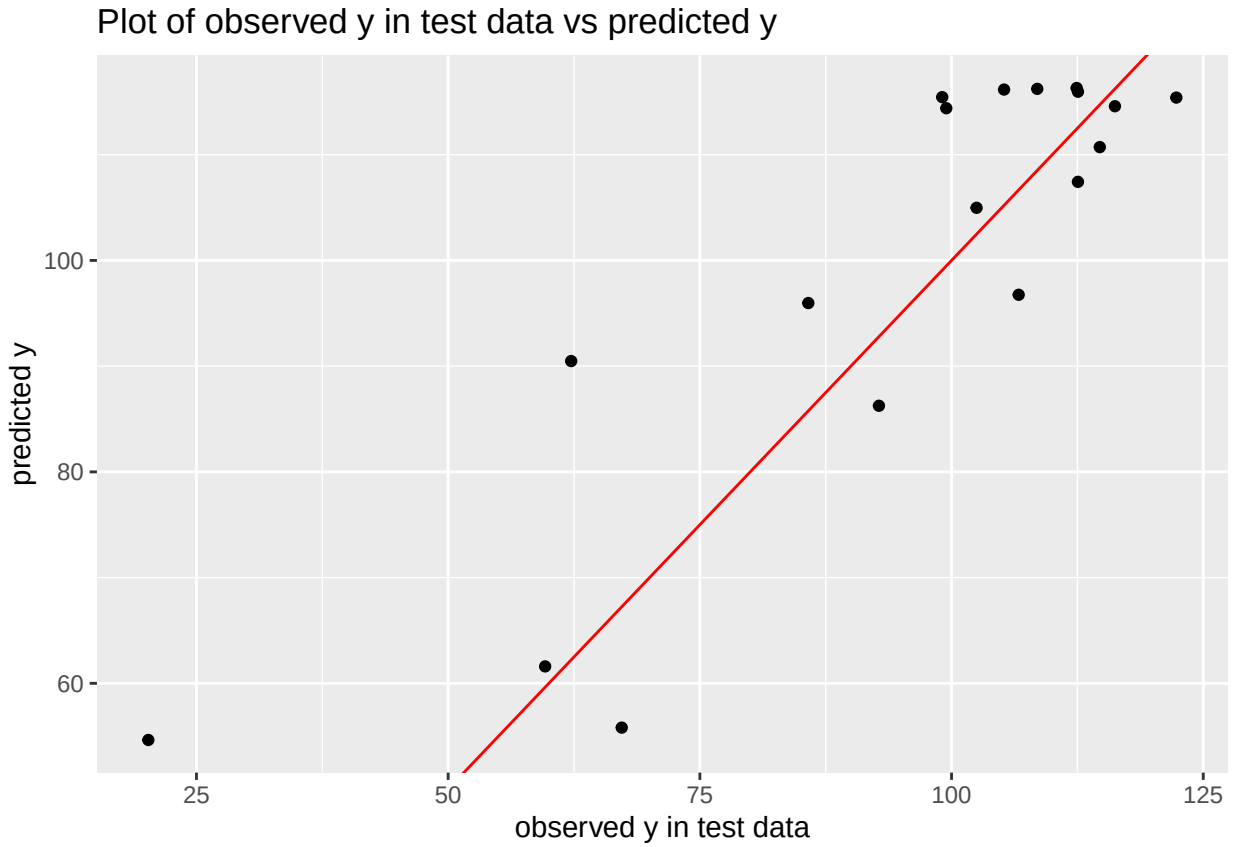
The fitted curve to some extent fits the data well and also few points are far away from the curve.

Apply the better model to the test set.

```
y_hat <- predict(reduced_bone_jaw_model, newdata = test_jawdata)
```

Plot the observed y values in test set versus their predicted values together with the reference line y = x, to check if the prediction seems reasonable.

```
y_test_jawdata <- test_jawdata$bone
ggplot( mapping = aes(x =y_test_jawdata , y = y_hat)) +
  geom_point()+
  geom_abline(intercept = 0, slope = 1, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```



Comment

Most of the points are far away from the reference line which indicates that the predicted values are significantly different from the observed values in the test data.

Computing the prediction mean square error (MSE)

```
prediction_MSE <- mean((y_test_jawdata - y_hat)^2)
prediction_MSE
```

```
## [1] 175.4818
```

The prediction mean square error (MSE) is 175.4818

4 Local regression methods

4.1 KNN regression model

V-FOLD CV FOR SELECTING K

```
set.seed(123)
library("FNN")
SSEP <- function(yobs, yhat) sum((yobs-yhat)^2)

K <- 2:15
V <- 6
# id.fold <- sample(1:V, size = NROW(train_jawdata), replace=T)
SSE <- rep(0, length(K))
for(k in 1:length(K)){
  id.fold <- sample(rep(1:V, each=trunc(NROW(train_jawdata)/V)))
  for(v in 1:V){
    train1<- train_jawdata[id.fold!=v, ];
    train2<- train_jawdata[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,]   2  1237.925
##  [2,]   3  1942.520
##  [3,]   4  1959.300
##  [4,]   5  4394.168
##  [5,]   6  5853.002
##  [6,]   7  5214.413
##  [7,]   8  8093.330
##  [8,]   9  9357.536
##  [9,]  10  9879.312
## [10,]  11 12595.400
## [11,]  12 15996.736
## [12,]  13 16297.677
## [13,]  14 18875.533
## [14,]  15 21616.916
```

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

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

Comment

To avoid over-fitting(that is when K=1), I choose K to range from 2 to 15 and using the 6-fold CV, we see that K=2, gives the optimal K.

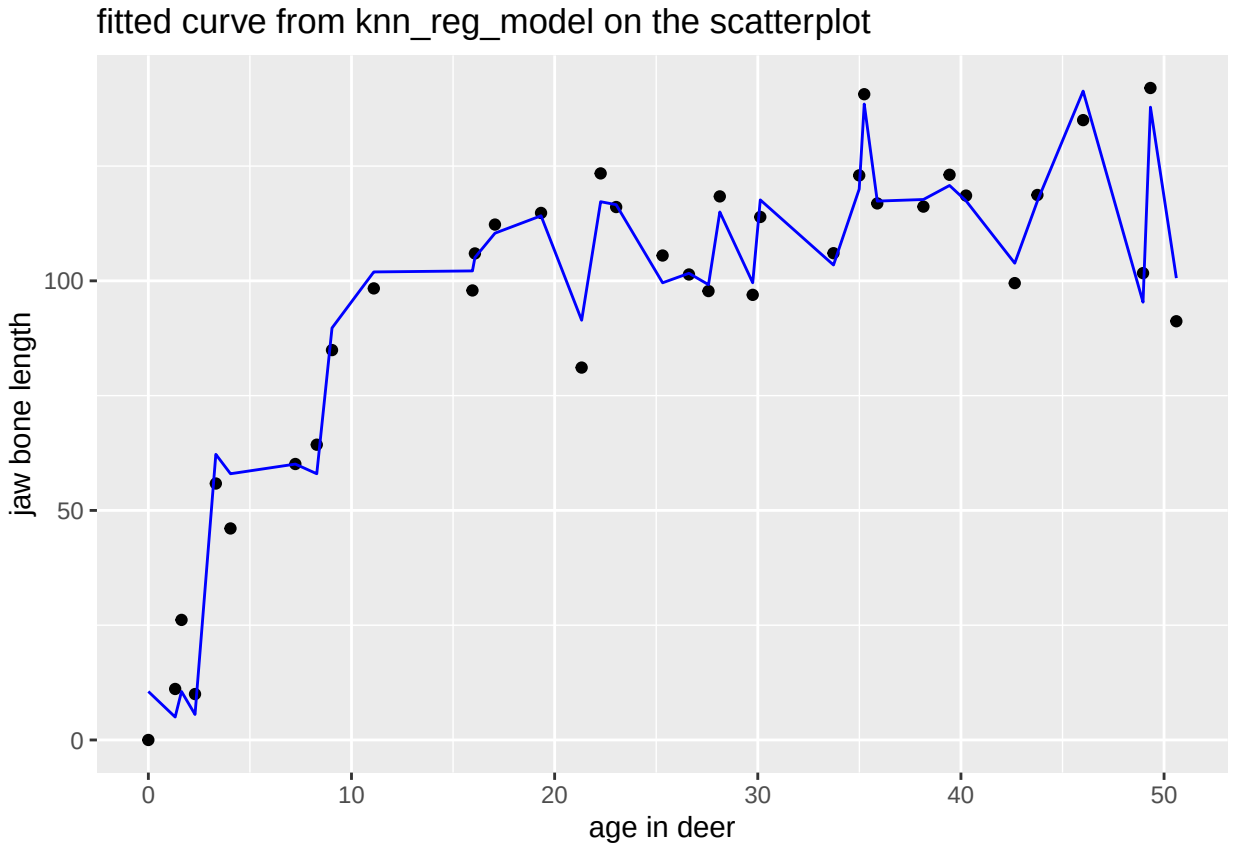
```
knn_reg_model <- knn.reg(train=train_jawdata, y=train_jawdata$bone,
                         k=k.opt, algorithm="kd_tree")
summary(knn_reg_model)
```

```
##           Length Class  Mode
## call       5     -none- call
## k          1     -none- numeric
## n          1     -none- numeric
## pred      36     -none- numeric
## residuals 36     -none- numeric
## PRESS      1     -none- numeric
## R2Pred     1     -none- numeric
```

```
#names(knn_reg_model)
```

Plot the fitted curve together with the scatterplot of the data.

```
library(ggplot2)
ggplot(data = train_jawdata, mapping = aes(x = age, y = bone)) +
  geom_point()+
  geom_line(aes(x=age, y=knn_reg_model$pred), col = 'blue') +
  ggtitle("fitted curve from knn_reg_model on the scatterplot") +
  xlab("age in deer") + ylab("jaw bone length")
```



Comment

The plot seems to fit the data very well. As wee can see the curve almost wiggles through all the data points.

```
knn_reg_model2 <- knn.reg(train=train_jawdata, test = test_jawdata,
                          y=train_jawdata$bone, k=k.opt, algorithm="kd_tree")
```

Plot the observed and predicted response values with reference line y = x

```
y_test_jawdata <- test_jawdata$bone
library(ggplot2)
ggplot( mapping = aes(x =y_test_jawdata , y = knn_reg_model2$pred)) +
  geom_point()+
  geom_abline(intercept = 0, slope = 1, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```

Comment

We observe that the points are near to the reference line which indicates that the observed values in thee test data is significant not different from the predicted values. This indicates a good prediction.

Computing the prediction mean square error (MSE)

```
prediction_MSE1 <-mean((y_test_jawdata - knn_reg_model2$pred)^2)
prediction_MSE1
```

```
## [1] 12.59012
```

Comment

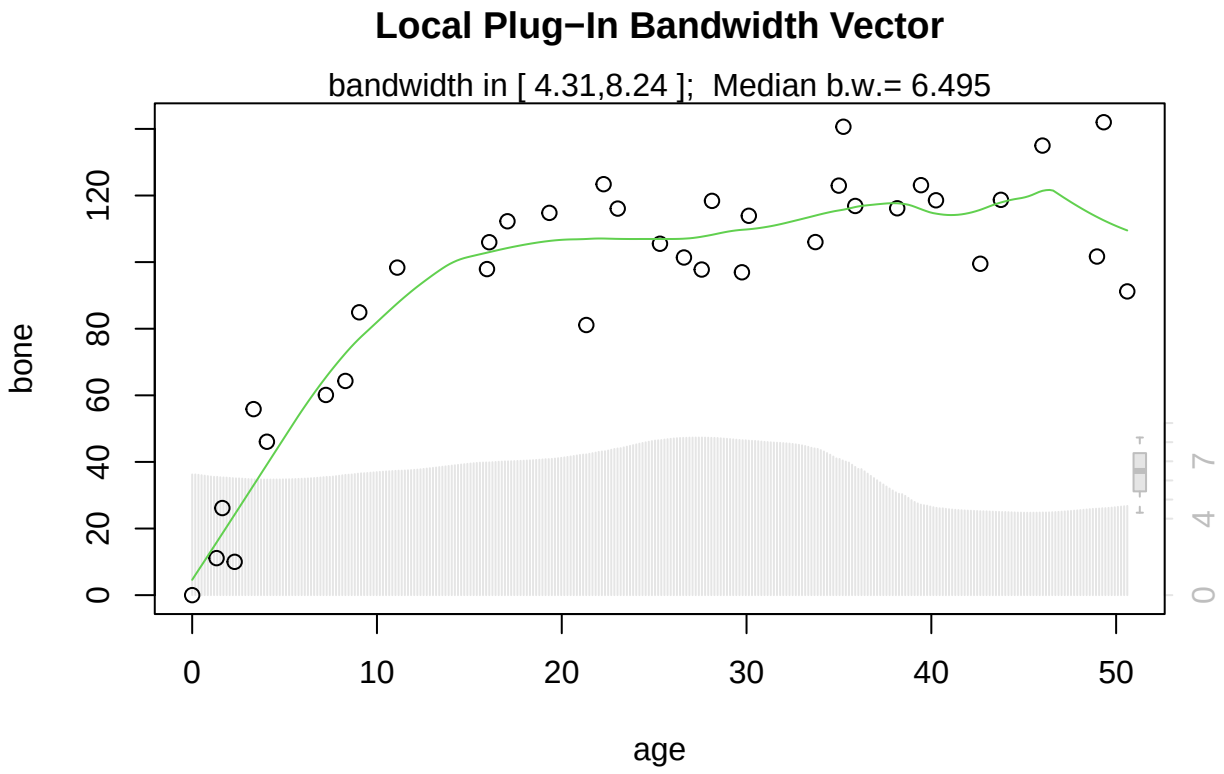
The prediction mean square error (MSE) is 12.59012 which is small which indicates the error of prediction is minimized.

4.2 Applying kernel regression to obtain a nonlinear fit.

```
library(lokern)
lofit <- lokerns(train_jawdata$age, train_jawdata$bone)
sb <- summary(lofit$bandwidth)

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 = train_jawdata, 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"])))
```



Comment

- We used Kernel Regression Smoothing with Local Plug-in Bandwidth.
- The bandwidth is within the interval [4.31,8.24] with median bandwidth of 6.495.

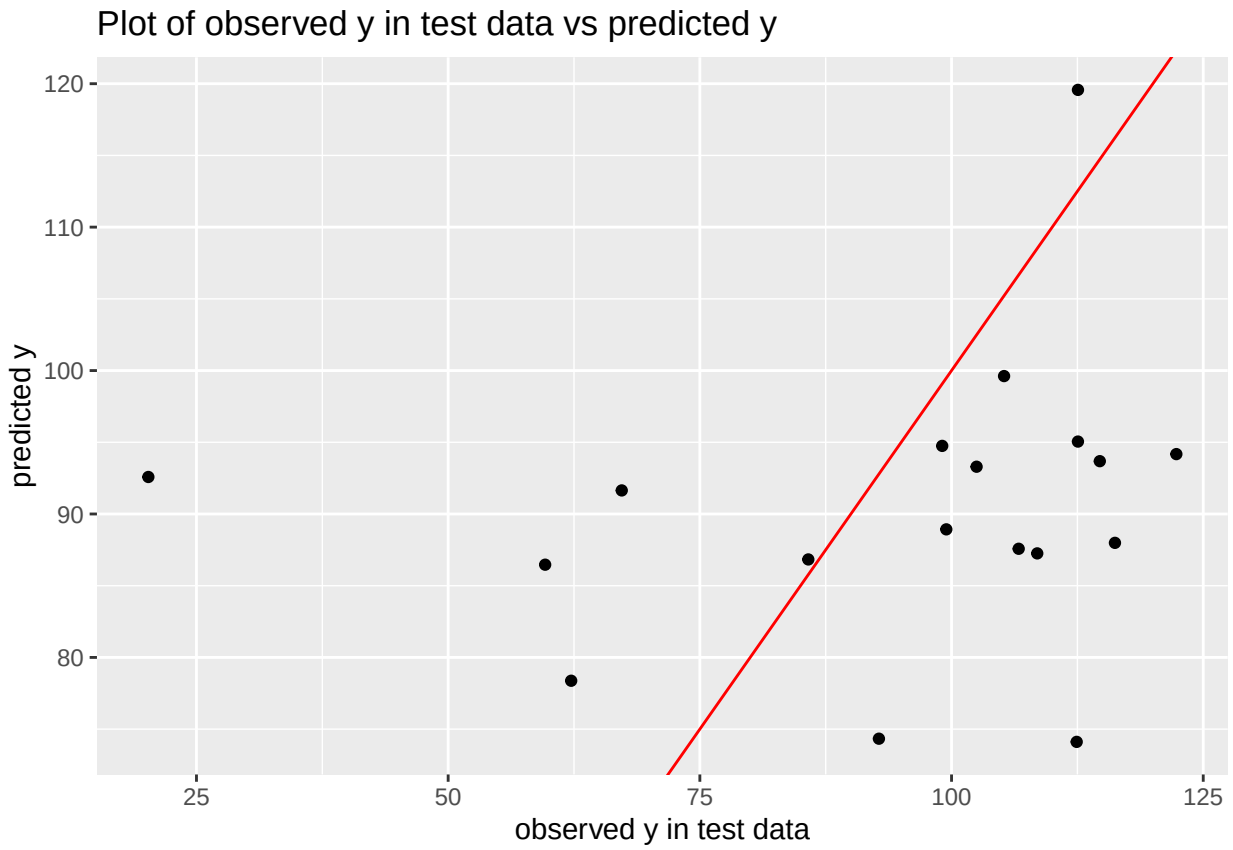
Apply the fitted kernel regression model to the test data.

```
perdict_kernel_reg <- predict(lofit, newdata = test_jawdata)
```

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

Plot the observed and predicted response values with reference line y = x

```
library(ggplot2)
ggplot( mapping = aes(x =y_test_jawdata , y = perdict_kernel_reg$y)) +
  geom_point()+
  geom_abline(intercept = 0, slope = 1, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```



Comment

We see from the plot that most of the points are far from the reference line which indicates poor prediction. That is the predicted values are statistically different from the observed values in the test data.

Computing the prediction MSE

```
prediction_MSE2 <- mean((y_test_jawdata - perdict_kernel_reg$y)^2)
prediction_MSE2
```

```
## [1] 670.7431
```

Comment

The prediction MSE is 670.7431 which is large.

4.3 local (cubic) polynomial regression

```
library(locpol)
fit.local <- locpol(bone~age, data=train_jawdata, deg=3, kernel=EpaK,bw =4)
```

Comment

- The kernel used here is EpaK
- The bandwidth used is 4

Apply the local cubic regression model to the test data.

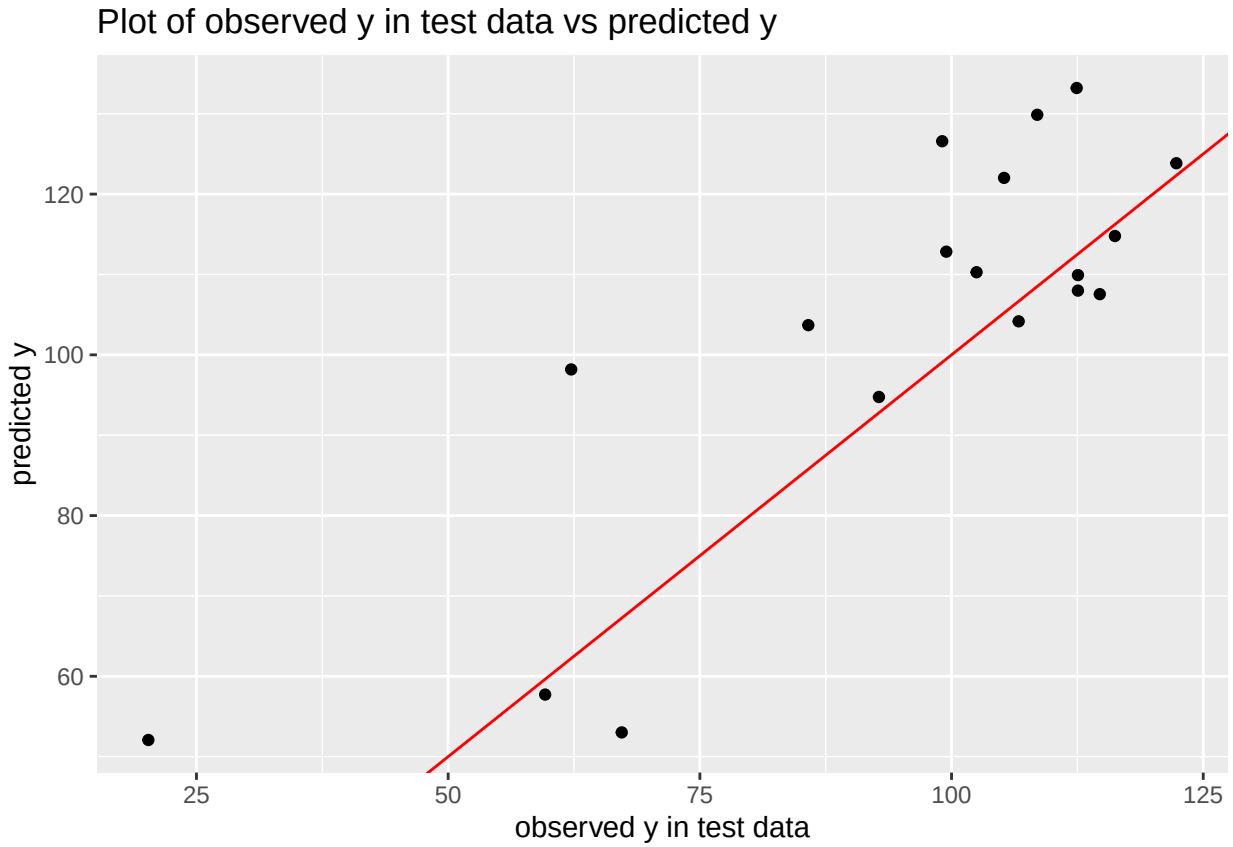
```
lp <- locpol(bone~age, data=train_jawdata, xeval=test_jawdata$age, deg=3,
             kernel=EpaK,bw =7)
lpp <- lp$lpFit
perdict_locpol <- lpp$bone
```

Comment

Here, I used bandwidth of 7 since I couldnt run with a bandwidth of 4.

Plot the observed and predicted response values with reference line y = x

```
library(ggplot2)
ggplot( mapping = aes(x =y_test_jawdata , y = perdict_locpol)) +
  geom_point()+
  geom_abline(intercept = 0, slope = 1, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```



Comment

We see from the plot that most of the points are far from the reference line which indicates poor prediction. That is the predicted values are statistically different from the observed values in the test data.

Computing Prediction MSE.

```
y_test_jawdata <- test_jawdata$bone
prediction_MSE3 <- mean((y_test_jawdata - perdict_locpol)^2)
prediction_MSE3
```

```
## [1] 283.0541
```

Comment

The prediction MSE is 283.0541 which is relatively large.

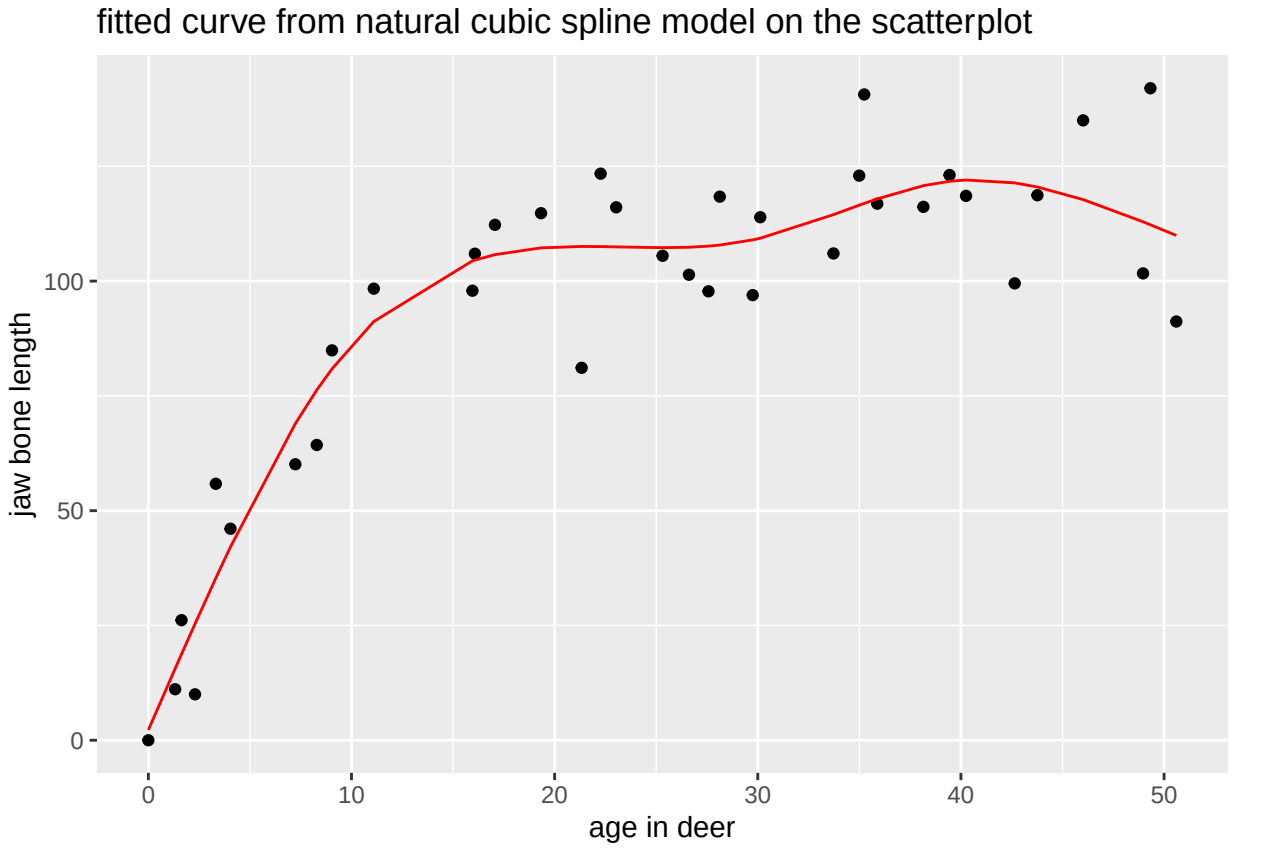
5 Regression/smoothing splines

5.1 Regression splines(Natural cubic splines)

```
library(splines)
#ns(train_jawdata$age, df = 5)
natural_cubspl <- lm(bone ~ ns(age, df = 5), data = train_jawdata)
```

Plot the resultant curve.

```
library(ggplot2)
ggplot(data = train_jawdata, mapping = aes(x = age, y = bone)) +
  geom_point()+
  geom_line(aes(x=age, y=natural_cubspl$fitted.values), col = 'red') +
  ggtitle("fitted curve from natural cubic spline model on the scatterplot") +
  xlab("age in deer") + ylab("jaw bone length")
```



Comment

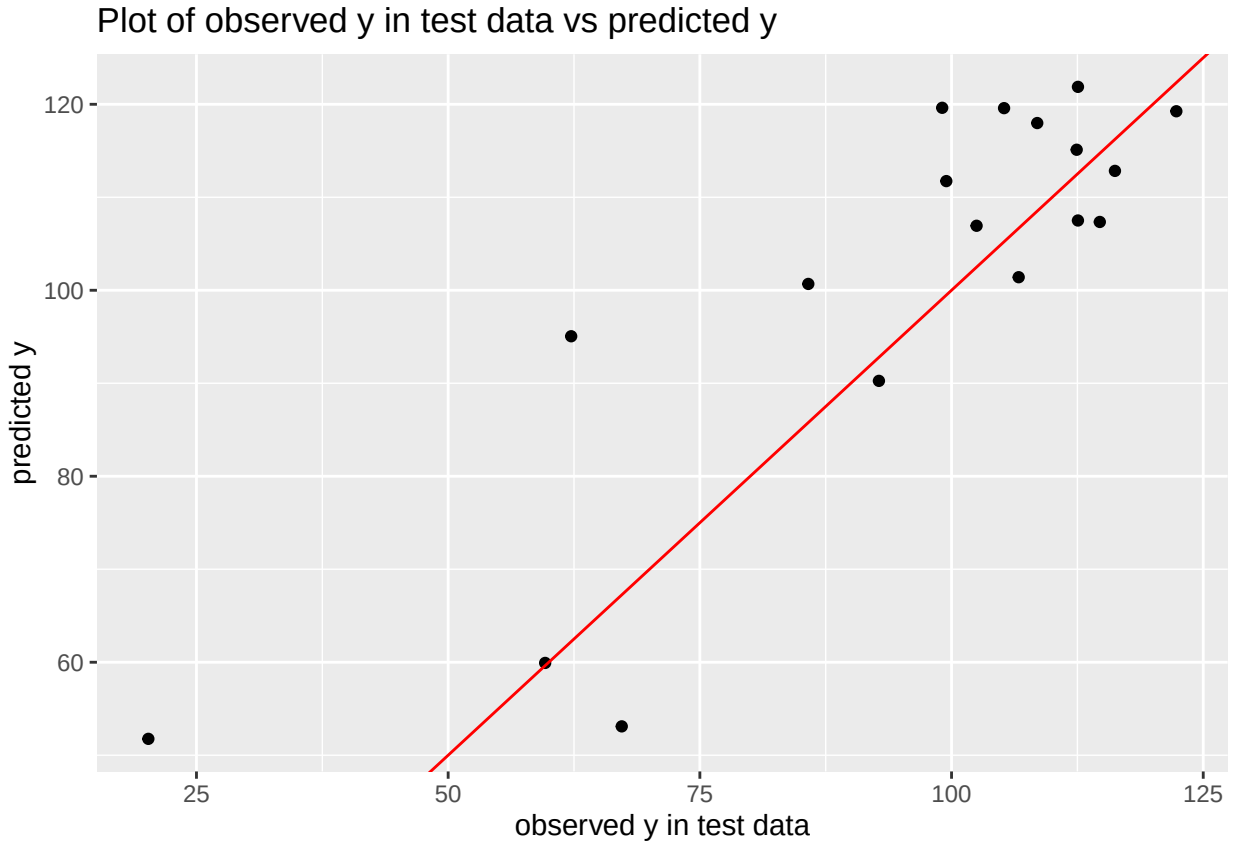
The curve does not fit the data very well. Most points are not near the curve.

Applying the fitted model to predict the test data.

```
y_hatt <- predict(natural_cubspl, newdata = test_jawdata)
```

Plot the observed and predicted response values with reference line y=x.

```
library(ggplot2)
ggplot( mapping = aes(x =test_jawdata$bone , y = y_hatt)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```



Comment

We see from the plot that most of the points are far from the reference line which indicates poor prediction. That is the predicted values are statistically different from the observed values in the test data.

Computing the prediction MSE

```
prediction_MSE4 <- mean((test_jawdata$bone - y_hatt)^2)
prediction_MSE4
```

```
## [1] 200.7106
```

Comment

The prediction MSE is 200.7106 which relatively large

5.2 Smoothing splines

```
library(splines)
smooth_spline <- smooth.spline(train_jawdata$age, train_jawdata$bone)
smooth_spline
```

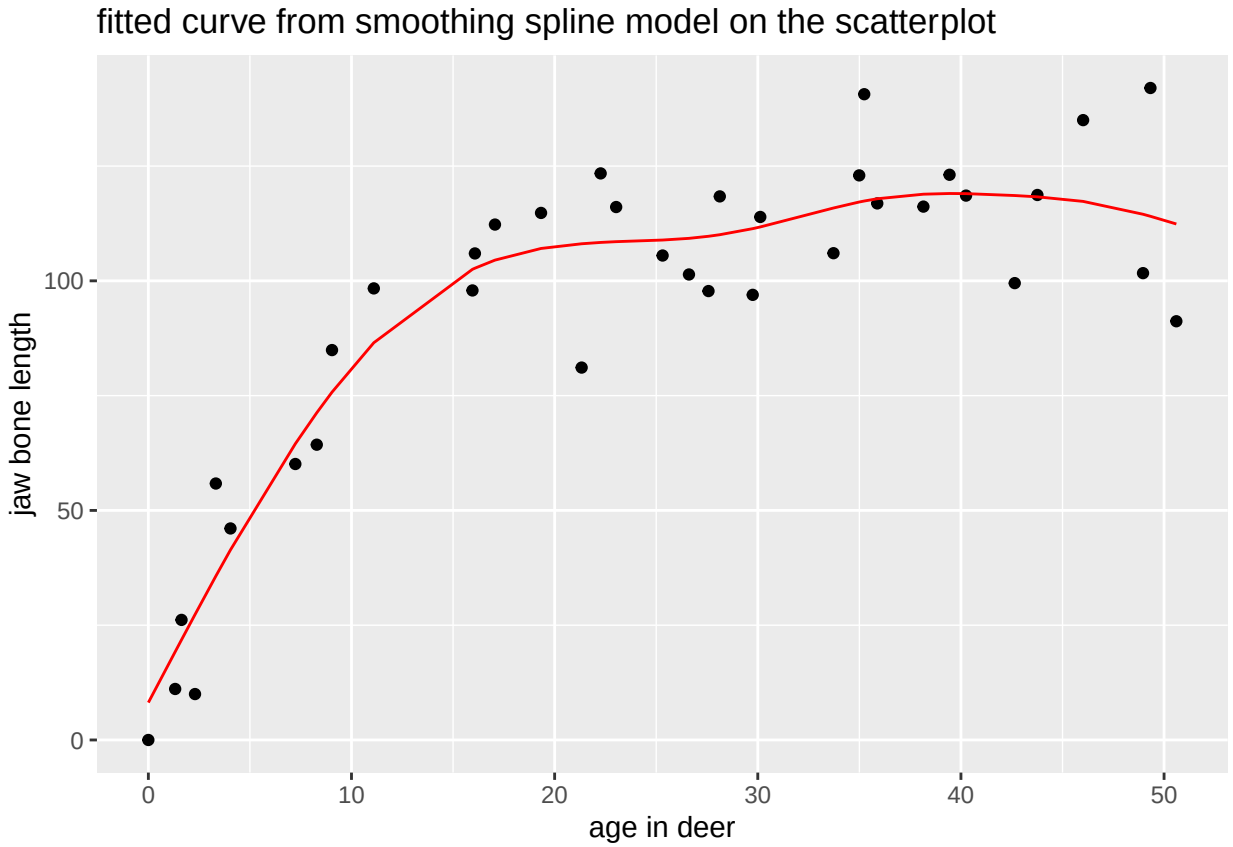
```
## Call:
## smooth.spline(x = train_jawdata$age, y = train_jawdata$bone)
##
## Smoothing Parameter  spar= 0.7339317  lambda= 0.001398752 (12 iterations)
## Equivalent Degrees of Freedom (Df): 5.508976
## Penalized Criterion (RSS): 5598.421
## GCV: 3892.639
```

Comment

The generalized cross-validation (GCV) was used for the smoothing parameter estimation. We obtained GCV of 3892.639 and an Equivalent Degrees of Freedom (Df) of 5.508976.

Add the resultant curve to the scatterplot.

```
library(ggplot2)
ggplot(data = train_jawdata, mapping = aes(x = age, y = bone)) +
  geom_point()+
  geom_line(aes(x=age, y=fitted(smooth_spline)), col = 'red') +
  ggtitle("fitted curve from smoothing spline model on the scatterplot") +
  xlab("age in deer") + ylab("jaw bone length")
```



Comment

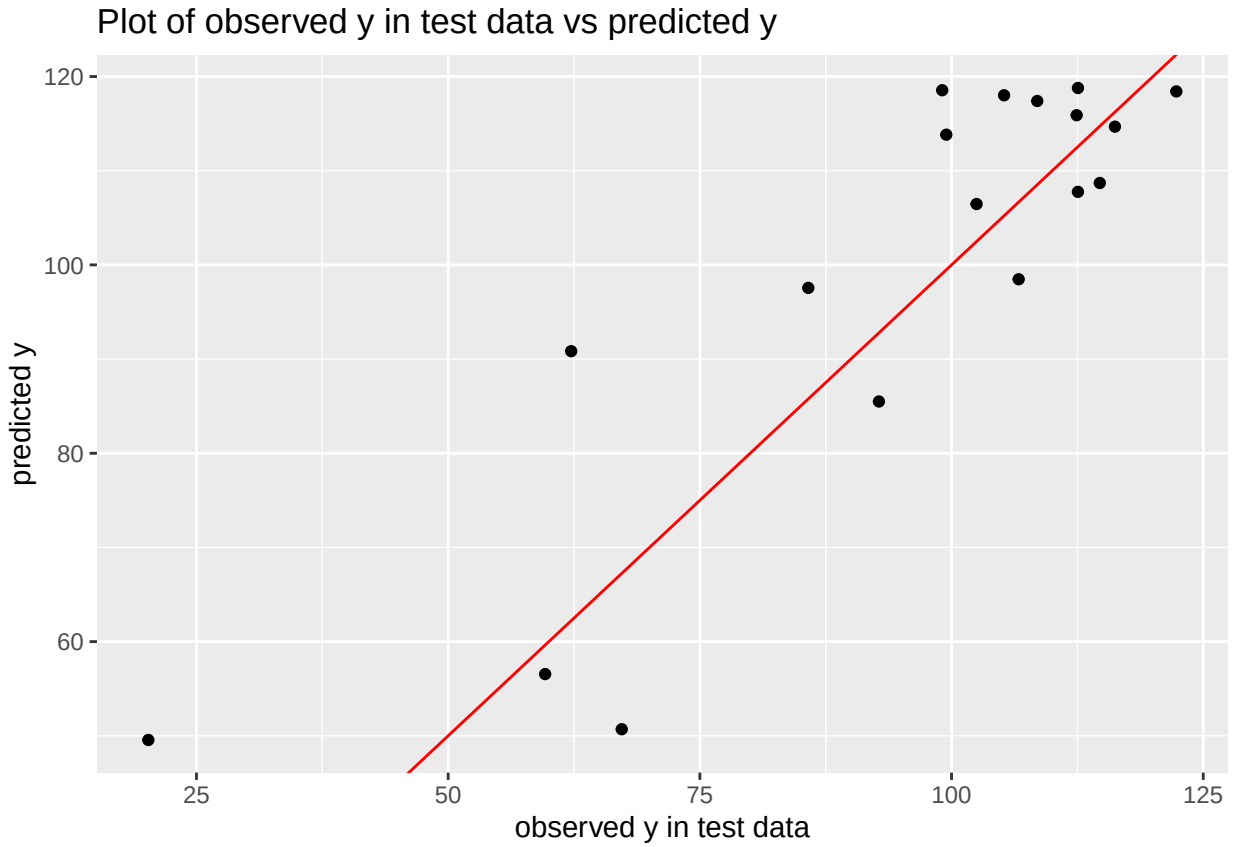
The curve does not fit the data very well. Most points are not near the curve.

Apply the fitted model to the test data.

```
y_hat_sp <- predict(smooth_spline, test_jawdata$age)
```

Plot the observed and predicted response values

```
library(ggplot2)
ggplot( mapping = aes(x =test_jawdata$bone , y = y_hat_sp$y)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  ggtitle("Plot of observed y in test data vs predicted y") +
  xlab("observed y in test data") + ylab("predicted y")
```



Comment

From the bottom to the middle part of the reference line we see that the points are not near the line. In particular the predicted values are statistically different from the observed values in the test data.

Compute the Prediction MSE

```
prediction_MSE5 <- mean((test_jawdata$bone - y_hat_sp$y)^2)
prediction_MSE5
```

```
## [1] 177.413
```

Comment

The prediction MSE is 177.413 which is relatively large.

6 Tabulate all the prediction MSE measures

```
measure <- c(prediction_MSE, prediction_MSE1,
             prediction_MSE2,prediction_MSE3,
             prediction_MSE4,prediction_MSE5)
method_measure <- data.frame("Methods"= c("Asymptotic exponential model",
                                "KNN regression","Kernel regression",
                                "Local cubic polynomial","Natural cubic spline",
                                "Smoothing Splines"), "Prediction_MSE"= measure)
knitr::kable(method_measure, align = "lc",
             caption = "A table of all the prediction MSE measures.")
```

Table 2: A table of all the prediction MSE measures.

Methods	Prediction_MSE
Asymptotic exponential model	175.48176
KNN regression	12.59012
Kernel regression	670.74305
Local cubic polynomial	283.05406
Natural cubic spline	200.71057
Smoothing Splines	177.41302

From the output of the prediction MSE for various methods, we see that KNN regression has the smallest prediction MSE. Hence, KNN regression gives the favorable result.