Project Four for Statistical Data Mining

Raymond Anthony Ford[*]

Due: 5 November 2018

Contents

1 Obtaining the data

We begin this project by bringing the data into R, and then plotting the data in a scatterplot to see if we notice any trends.

```r
dat <- read.csv("https://sites.google.com/site/utepstat5494/file-cabinet/jaws.txt",
                header=TRUE, sep="\t")
plot(dat$age, dat$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers")
```

In Figure 1 we see that the relationship between age and jaw bone length is nonlinear, thus we will need to use some nonlinear methods.

We next split our data into two parts: a training data set and a testing data set. This is a accomplished with the code below.

[*]raford2@miners.utep.edu

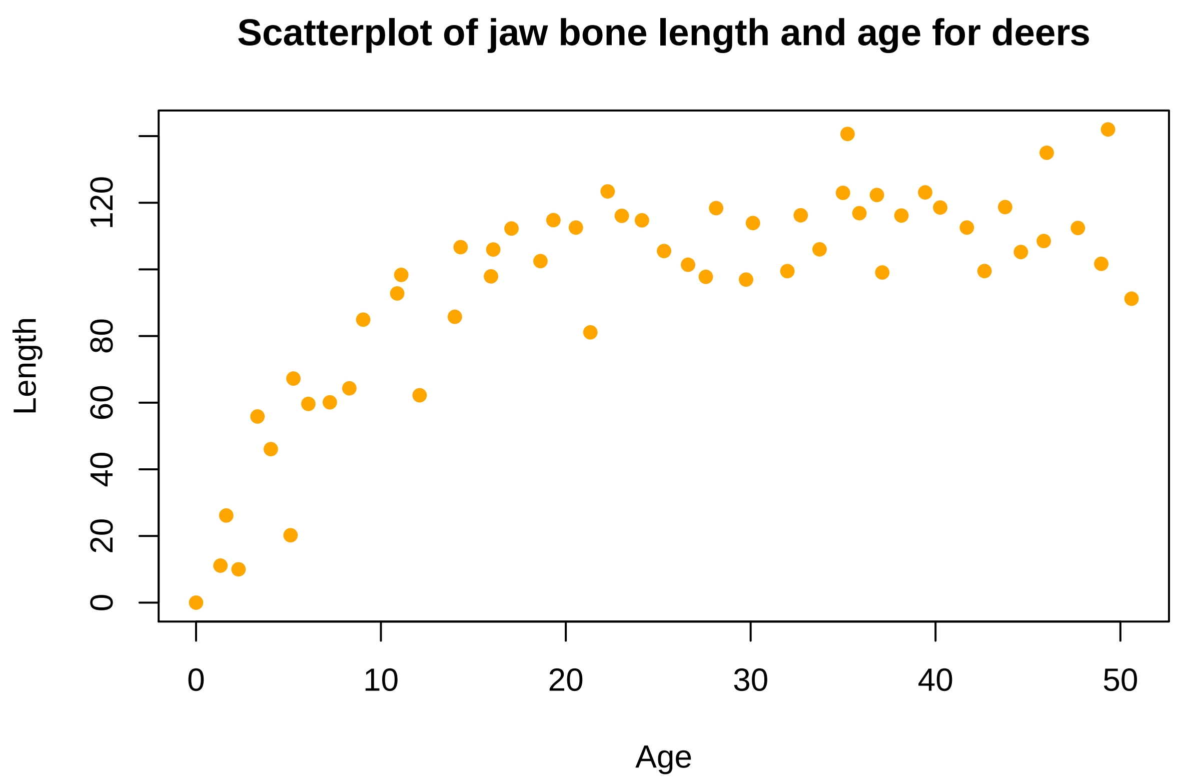

Figure 1: Scatter of the deer jaw measurement data set.

```r
set.seed(5494)
training.data.index <- sample(1:nrow(dat), 0.667*nrow(dat))
training <- dat[training.data.index, ]
test <- dat[-training.data.index, ]
```

2 Parametric nonlinear models

The first model that we will fit is an asymptotic exponential model of the following form.

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

We fit this model and then provide a summary of this model with the following code.

```r
aexp.mod <- nls(bone ~ beta1 - beta2*exp(-beta3*age), data=training,
    start=list(beta1 = 125, beta2 = 50, beta3 = 1), trace=FALSE)
summary(aexp.mod)
```

```
##
## Formula: bone ~ beta1 - beta2 * exp(-beta3 * age)
##
## Parameters:
```

```
##         Estimate Std. Error t value Pr(>|t|)    
## beta1 116.98493    4.25134  27.517  < 2e-16 ***
## beta2 119.69323    9.39485  12.740 2.73e-14 ***
## beta3   0.11765    0.02023   5.816 1.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 15.03 on 33 degrees of freedom
## 
## Number of iterations to convergence: 6 
## Achieved convergence tolerance: 4.089e-07
```

In the above summary output we note that β_1 and β_2 only differ by three numbers.

We are instructed to test $H_0 : \beta_1 = \beta_2$. To do that we must first construct the reduced model. The code used to build this reduced model is below.

```
aexp.mod.red <- nls(bone ~ beta1 - beta1*exp(-beta3*age), data=training,
    start=list(beta1 = 125, beta3 = 1), trace=FALSE)
```

To test this hypothesis we perform an ANOVA. The code for this test and it's ouput is below.

```
anova(aexp.mod, aexp.mod.red)
```

```
## Analysis of Variance Table
## 
## Model 1: bone ~ beta1 - beta2 * exp(-beta3 * age)
## Model 2: bone ~ beta1 - beta1 * exp(-beta3 * age)
##   Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
## 1     33     7456.7                          
## 2     34     7475.6 -1 -18.862  0.0835 0.7744
```

From the ANOVA output we see that we obtain a p-value of 0.77 and conclude that we fail to reject our null hypothesis, thus it would be better to use the reduced model.

Next, we plot the training data along with the fitted curve for the model in the following scatterplot.

```
better.model <- aexp.mod.red
plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
     sub="Asymptotic exponential model")
lines(sort(training$age), fitted.values(better.model)[order(training$age)],
      col="blue", lty=2, lwd=2)
```

In Figure 2 we note that while the curve seems to follow the data in the trend pretty well, there likely exists other methods that would contain more "wiggles" and thus fit the data better.

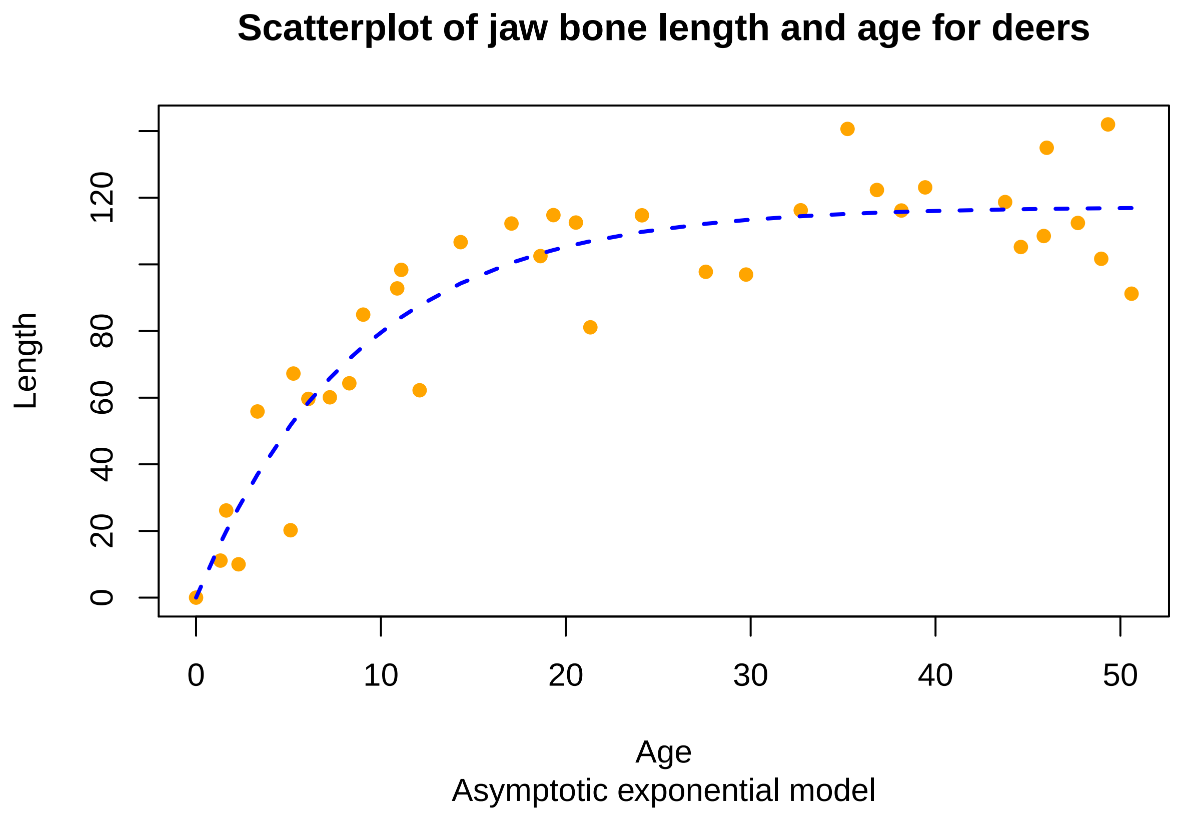

Figure 2: Results from parametric nonlinear least squares.

Finally, we compute the prediction mean square error using the testing data with the following code.

```r
pMSE.aexp.mod.red <- mean((test$bone - predict(aexp.mod.red, test))^2)
pMSE.aexp.mod.red
```

```
## [1] 84.12192
```

3 Local regression methods

3.1 KNN regression

We next fit a model using KNN with $k = 4$, and plot the training data along with the fitted curve for the model in the following scatterplot.

```r
library("FNN")
fit.knn <- knn.reg(train=training, y=training$bone, k=4, algorithm="kd_tree")
plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
     sub="kNN regression")
lines(sort(training$age), sort(fit.knn$pred), col="blue", lwd=2, lty=2)
```

In Figure 3 we see that the kNN model follows the overall trend of the data and the presence of more “wiggles” brings the fitted line closer to the actual data.

Finally, we compute the prediction mean square error using the testing data with the following code.

```r
yhat.knn <- knn.reg(train=training, y=training$bone, k=4, algorithm="kd_tree",
                    test=as.data.frame(test))
pMSE.knn <- mean((test$bone - yhat.knn$pred)^2)
pMSE.knn
```

```
## [1] 19.2222
```

3.2 Kernal regression

We next fit a model using kernal regression, and plot the training data along with the fitted curve for the model in the following scatterplot. I used the default setting for bandwidth from the `lokerns()` in `R`.

```r
library(lokern)
kern.fit <- lokerns(x=training$age, y=training$bone)
plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
```

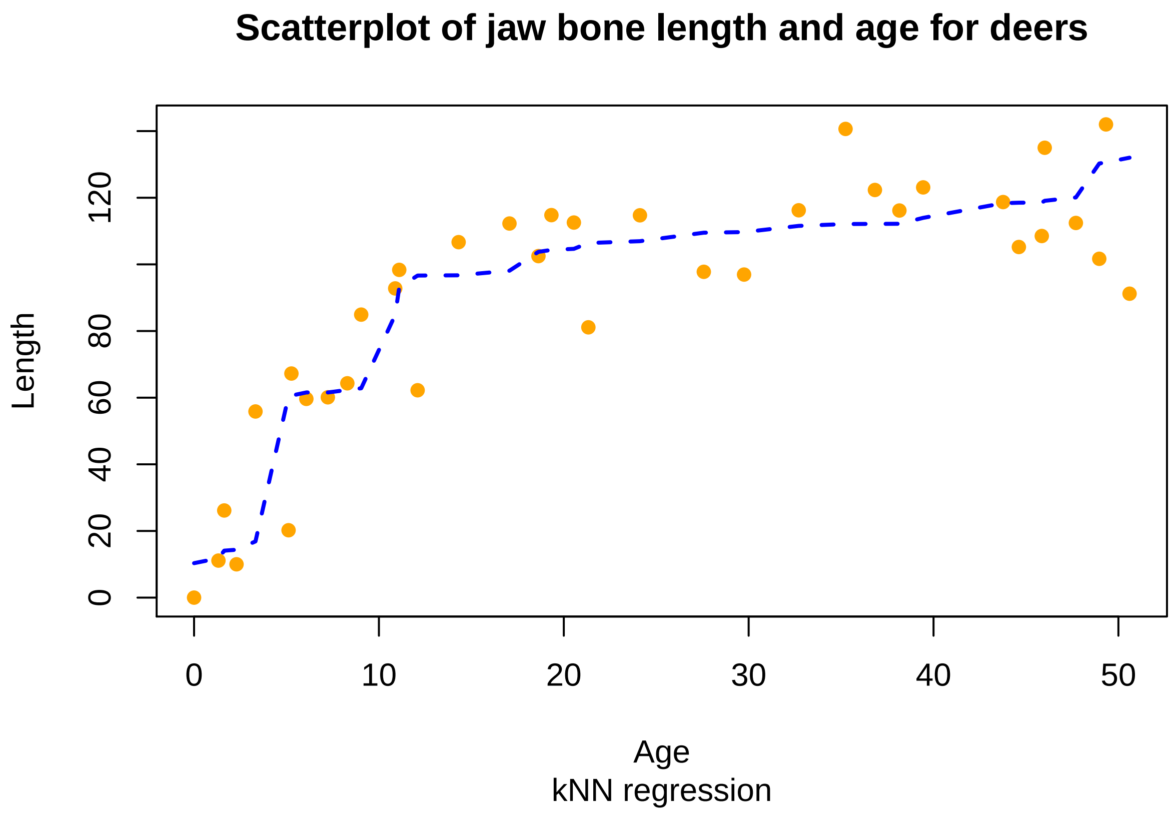

Figure 3: Results from knn regression.

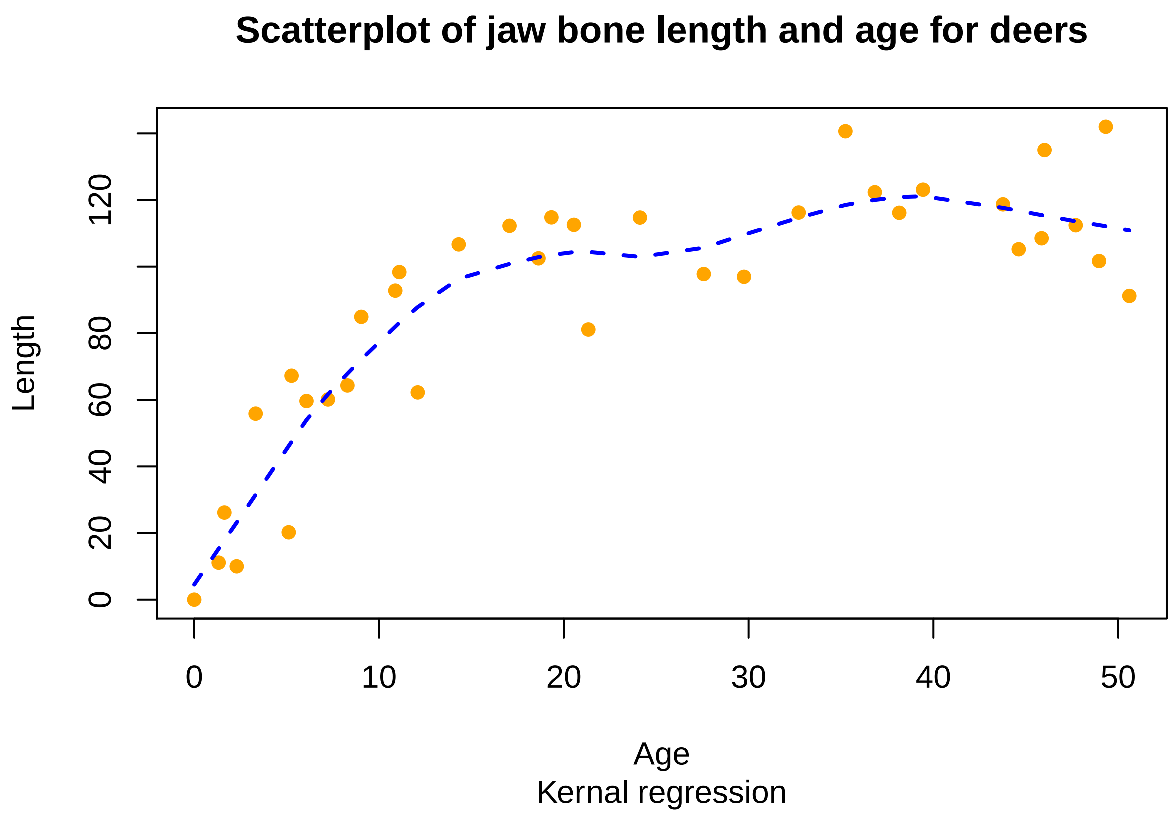

Figure 4: Results from kernal regression.

```
     sub="Kernal regression")
lines(sort(training$age), fitted.values(kern.fit)[order(training$age)],
      col="blue", lty=2, lwd=2)
```

In Figure 4 we see that the line continues to follow the overall trend in the training data, but it is more smoother that the one obtained from kNN (i.e. not as many "wiggles").

Finally, we compute the prediction mean square error using the testing data with the following code.

```
pMSE.kern <- mean((test$bone - predict(kern.fit, test$age)$y)^2)
pMSE.kern
```

```
## [1] 254.7341
```

3.3 Local polynomial regression

We next fit a model using local polynomial regression, and plot the training data along with the fitted curve for the model in the following scatterplot.

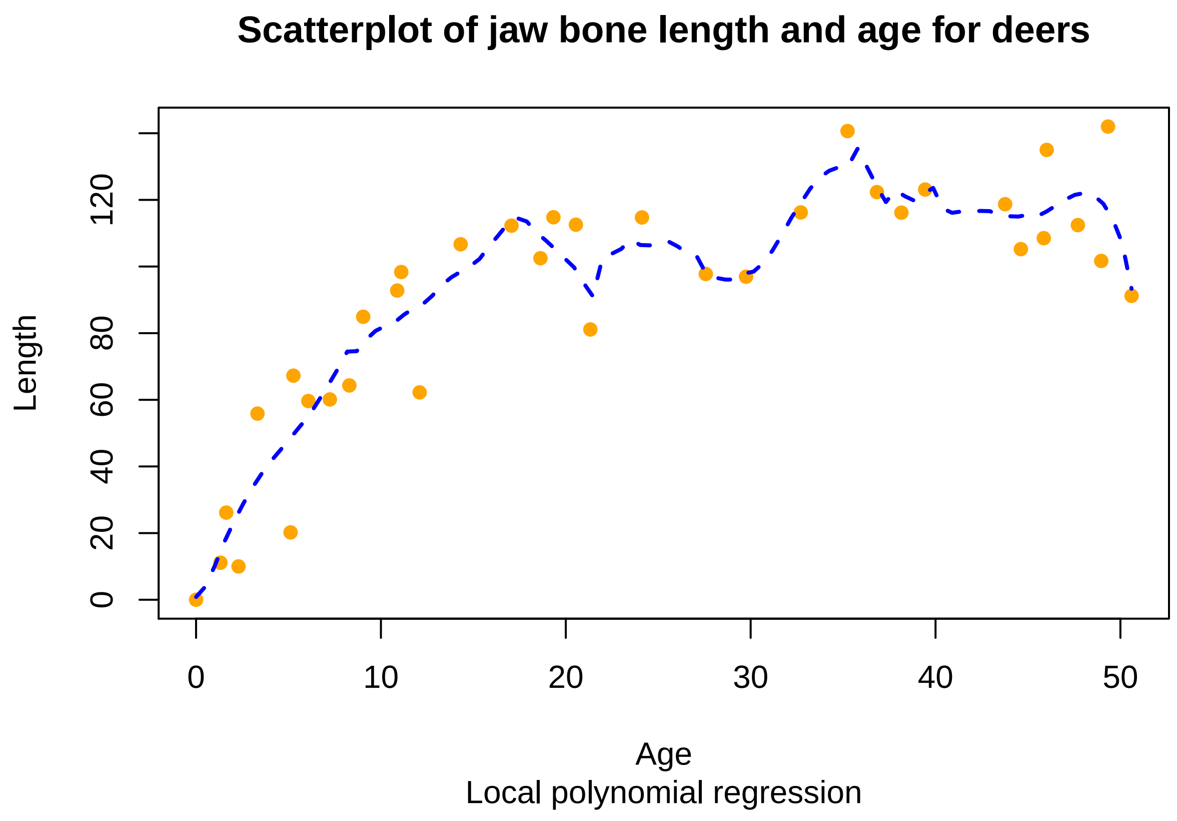

Figure 5: Results from local polynomial regression.

```
library(locpol)
localpoly.fit <- locpol(bone ~ age, data=training, deg=3, kernel=CosK, bw=6)
 plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
     sub="Local polynomial regression")
ages <- seq(from=min(training$age), to=max(training$age),
            length.out=length(fitted.values(localpoly.fit)))
lines(sort(ages), fitted.values(localpoly.fit), lwd=2, lty=2, col="blue")
```

In Figure 5 we see more "wiggles" in the line and the "cut the mountain, fill the valley" phenomenon that was discussed during lecture.

Finally, we compute the prediction mean square error using the testing data with the following code.

```
yhat.polyfit <- locpol(bone ~ age, data=training, deg=3,
                       kernel=CosK, bw=6, xeval=test$age)$lpFit$bone
pMSE.polyfit <- mean((test$bone - yhat.polyfit)^2)
pMSE.polyfit
```

```
## [1] 179.5543
```

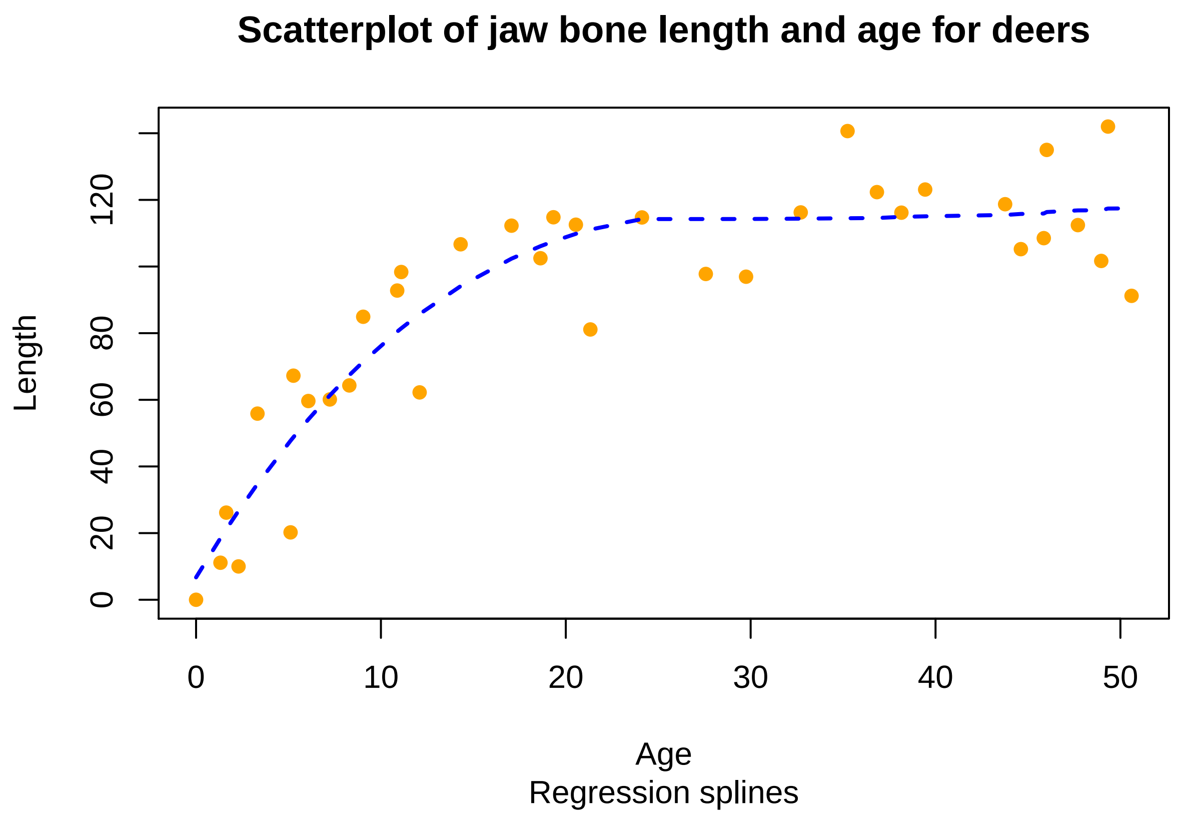

Figure 6: Results from regression splines.

4 Regression smoothing splines

4.1 Regression splines

We next fit a model using regression splines (natural cubic splines), and plot the training data along with the fitted curve for the model in the following scatterplot.

```
library(splines)
fit.spline <- lm(training$bone ~ bs(training$age, degree=3))
plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
     sub="Regression splines")
lines(sort(training$age), sort(predict(fit.spline, data.frame(training$age))),
      lty=2, col="blue", lwd=2)
```

In Figure 6 we see that while the line follows the overall trend of the data, it appears to be relatively far away from the points.

Finally, we compute the prediction mean square error using the testing data with the following code.

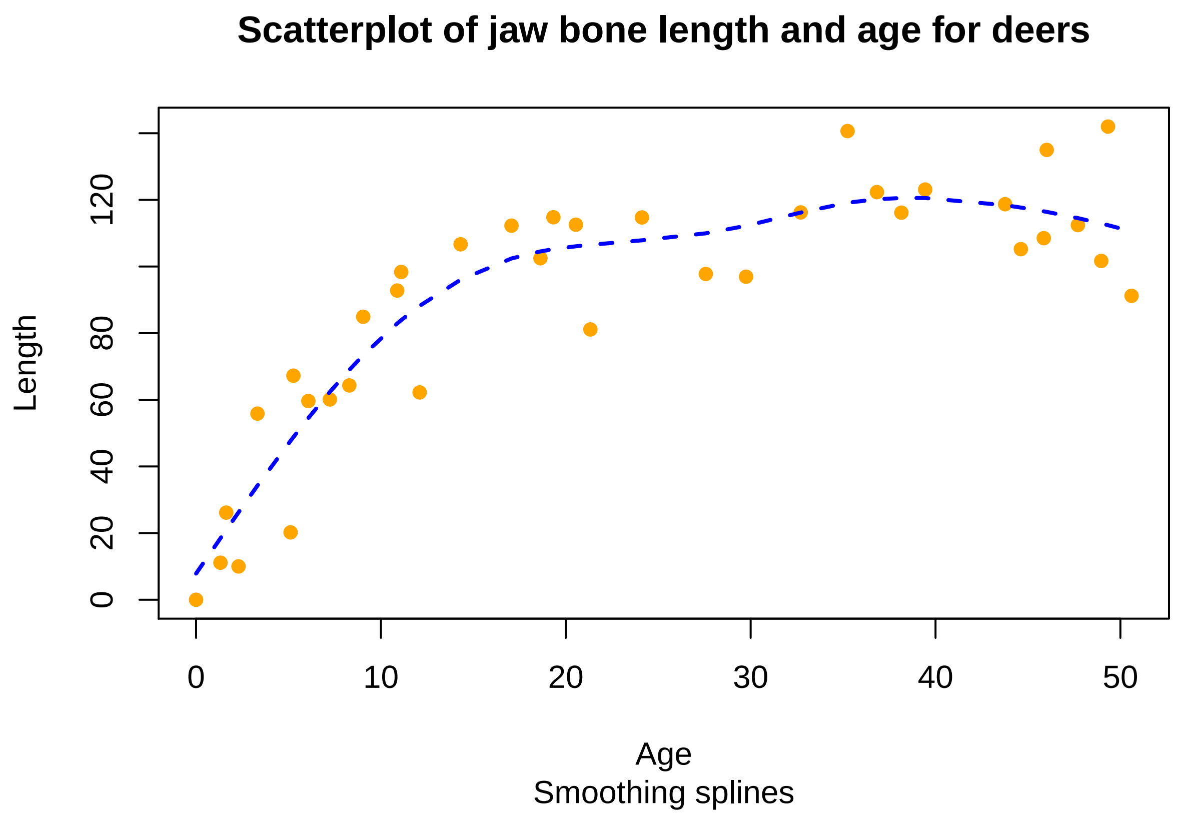

Figure 7: Results from smoothing splines.

```r
pMSE.regspl <- mean((test$bone - predict(fit.spline, age=data.frame(test$age)))^2)
pMSE.regspl
```

```
## [1] 1849.852
```

4.2 Smoothing splines

We finally fit a model using smoothing splines, and plot the training data along with the fitted curve for the model in the following scatterplot. Our tuning parameter is chosen via cross validation built-in to the `smooth.spline()` function.

```r
fit.smth <- smooth.spline(x=training$age, y=training$bone, cv=TRUE)
plot(training$age, training$bone, xlab="Age", ylab="Length", pch=16, col="orange",
     main="Scatterplot of jaw bone length and age for deers",
     sub="Smoothing splines")
lines(fit.smth, lty=2, lwd=2, col="blue")
```

In Figure 7 we see that the line follows the overall trend of the data, but similar to the regression splines the data tends to be relatively far away from the line.

Finally, we compute the prediction mean square error using the testing data with the following code.

```
pMSE.smth <- mean((test$bone - predict(fit.smth, test$age)$y)^2)
pMSE.smth
```

```
## [1] 107.8475
```

5 Comparing the methods

Finally we compare the performance of all six models with respect to prediction mean square error (MESP). This comparison can be seen below.

```
pmses <- c(pMSE.aexp.mod.red, pMSE.knn, pMSE.kern, pMSE.polyfit, pMSE.regspl,
           pMSE.smth)
pmses <- as.data.frame(t(t(pmses)))
pmses$er <- c("Aexp", "KNN", "Kernal", "Poly", "RegSpl", "SmthSpl")
names(pmses) <- c("MSEP", "Method")
pmses
```

```
##         MSEP  Method
## 1   84.12192    Aexp
## 2   19.22220     KNN
## 3  254.73410  Kernal
## 4  179.55429    Poly
## 5 1849.85154  RegSpl
## 6  107.84749 SmthSpl
```

From the output we see that the model that performed the best was kNN regression, and the model that performed the worst was the regression splines model. The model that performed second best was the asymptotic exponential model. I do wonder if we were provided this model because it arises in nature (i.e. did a subject matter expert come up with it?).