# Project05

Isaiah Thompson Ocansey

2022-11-08

1a) 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?

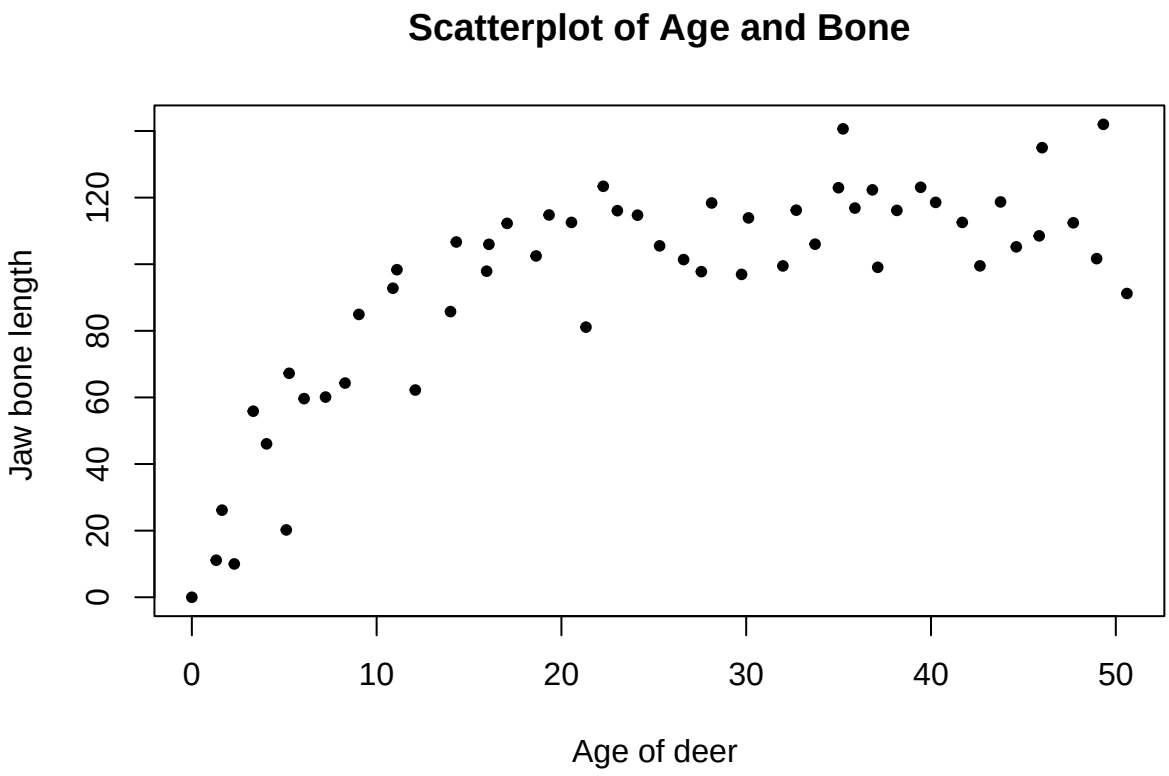
```
df <- read.table(file="jaws.txt", header = TRUE)
dim(df);head(df)
```

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

```
##         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 set has 54 observations and 2 variables*

```
plot(df$age, df$bone, main="Scatterplot of Age and Bone",
    xlab="Age of deer ", ylab="Jaw bone length ", pch=20)
```




*The above plot shows the scatter plot of the age of deer and the length of the jaw bones. It can be observed that there's a nonlinear relationship between the two predictors*

**DATA PARTIONING**

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

```
set.seed(123)
sample <- sample(nrow(df), (2/3)*nrow(df), replace = FALSE) # training set
D1 <- df[sample, ]
#test set
D2 <- df[-sample, ]
dim(D1); dim(D2)
```

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

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

*We randomly partition the data set into training and testing set in the ratio 2 is to 1 respectively and after the partition, the training set D1 has 36 observations and 2 variables whiles the testing set D2 has 18 observations and 2 variables*

(2b) 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.

```
min.max.df<-c(min(df$age),max(df$age),max(df$age)-min(df$age))
min.max.D1<-c(min(D1$age),max(D1$age),max(D1$age)-min(D1$age))
min.max.D2<-c(min(D2$age),max(D2$age),max(D2$age)-min(D2$age))
min.max.D1;min.max.D2
```

```
## [1]  1.32000 49.32956 48.00956
```

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

### PARAMETRIC NONLINEAR MODELS

(3a) Fit an asymptotic exponential model of the following form

```
attach(D1)
model1 <- nls(bone ~ beta1 - beta2*exp(-beta3*age),
start=list(beta1 = 100, beta2 = 90, beta3 = 0.3), trace=T)
```

```
## 12019.56    (1.08e+00): par = (100 90 0.3)
## 7218.668    (6.88e-01): par = (105.2541 88.83676 0.1157341)
## 4897.436    (2.21e-02): par = (114.2854 109.4733 0.1269187)
## 4895.049    (1.61e-04): par = (114.2167 110.0838 0.1267379)
## 4895.049    (2.02e-05): par = (114.2188 110.0775 0.1267163)
## 4895.049    (2.55e-06): par = (114.2191 110.0768 0.1267136)
```

```
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
```

*After fitting the asymptotic exponential model as shown above, we observe that all the coefficients are significant*

(3b) To test in Model (1), fit the reduced model, i.e., the model by plugging in the condition under H0 and use the anova function. Also, compare two nls models with AIC or BIC. Then conclude on which model is better.

```
model2 <- nls(bone ~ beta1*(1-exp(-beta3*age)),
  start=list(beta1 = 100, beta3 = 0.3), trace=T)
```

```
## 11009.56    (9.63e-01): par = (100 0.3)
## 6292.938    (5.19e-01): par = (104.9752 0.1705696)
## 4972.119    (1.09e-01): par = (113.0795 0.1290611)
## 4913.108    (3.94e-03): par = (113.8081 0.1333996)
## 4913.027    (2.56e-04): par = (113.7838 0.1337274)
## 4913.027    (1.72e-05): par = (113.7812 0.1337499)
## 4913.027    (1.15e-06): par = (113.781 0.1337514)
```

```
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: 1.147e-06
```

### COMPARING THE TWO MODELS USING ANOVA

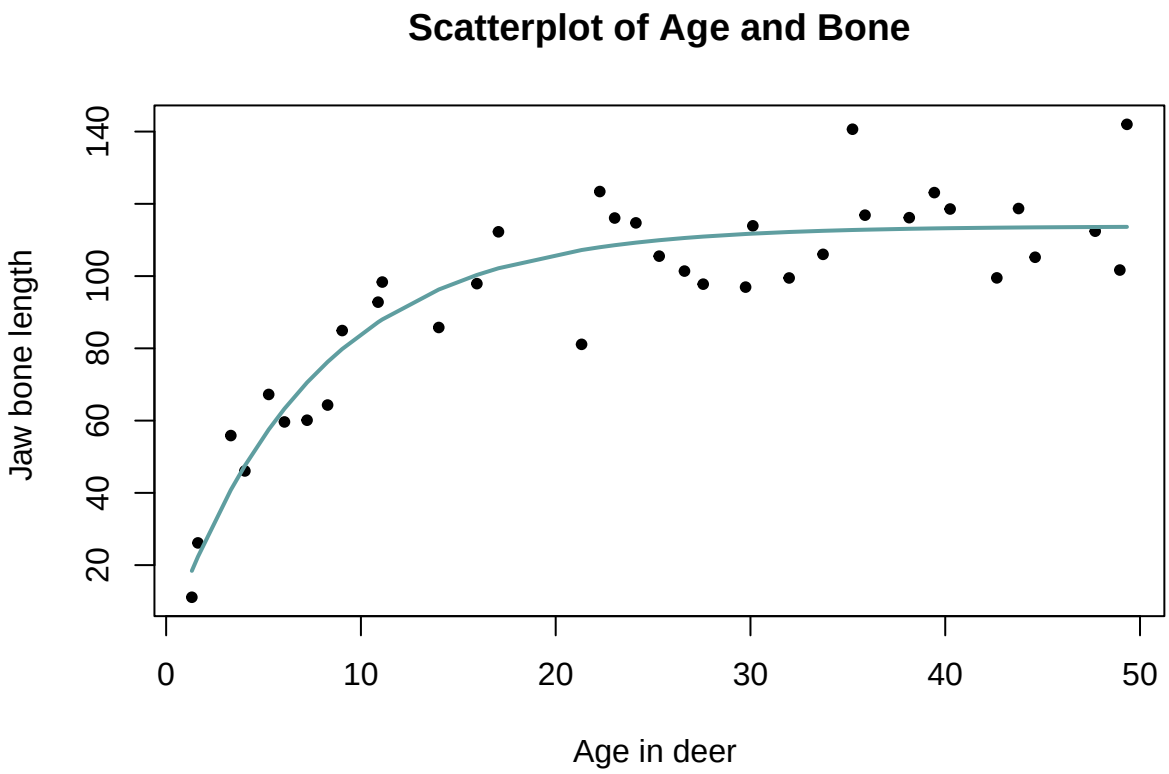
```
# Compare the two models
anova(model2,model1)
```

```
## 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
```

*From the above output, it can be observed that the p value of the test is 0.73 which shows the second model, model2 is not significantly different from the first model model1 at alpha equal to 0.05. Hence, we conclude that the second model, Model2 is better than model1*

(3c) Based on the better model in 2(b), add the fitted curve to the scatterplot.

```
# Fitted Curve vs Train Data
plot(D1$age, D1$bone, main="Scatterplot of Age and Bone",
    xlab="Age in deer ", ylab="Jaw bone length ", pch=20)
lines(sort(D1$age), fitted.values(model2)[order(D1$age)], lwd=2,col="cadetblue")
```



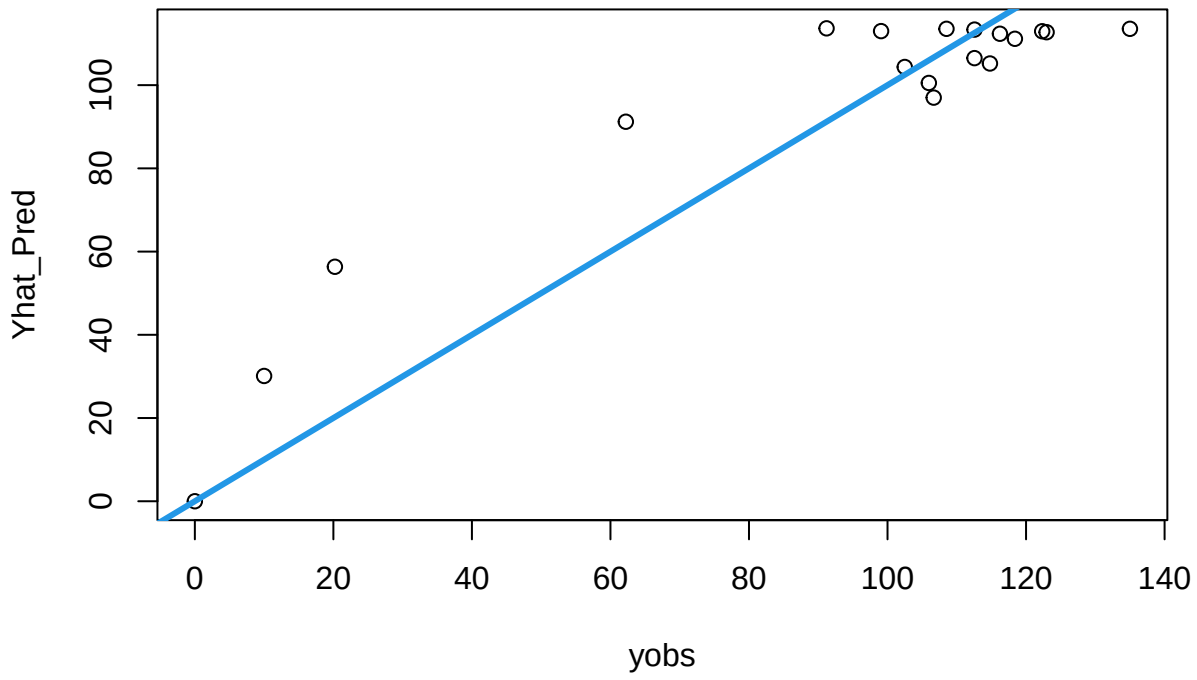

*The above plot shows the fitted curve based on model2*

(3d) Apply the better model in 2(b) to the test set D2. Plot the observed yi values in D2 versus their predicted values, together with the reference line , to check if the prediction seems reasonable. And computer the prediction mean square error (MSE)

```
# MEAN SQUARE ERROR FOR PREDICTION
Yhat_Pred<- predict(model2, newdata = D2); Yhat_Pred
```

```
##  [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
```

```
yobs <- D2[, 2]
plot(yobs,Yhat_Pred)
abline(a=0,b=1,col=4,lwd=3)
```




```
MSEP1 <- mean((yobs-Yhat_Pred)^2)
MSEP1
```

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

*When we apply the second model, model2 to the testing data set, we can observe a mean square error of prediction of 236.0823*

(4a) On basis of D1, obtain a KNN regression model with your choice of K. Plot the fitted curve together with the scatter plot of the data. Apply the fitted model to D2. Plot the observed and predicted response values with reference line and obtain the prediction MSE.

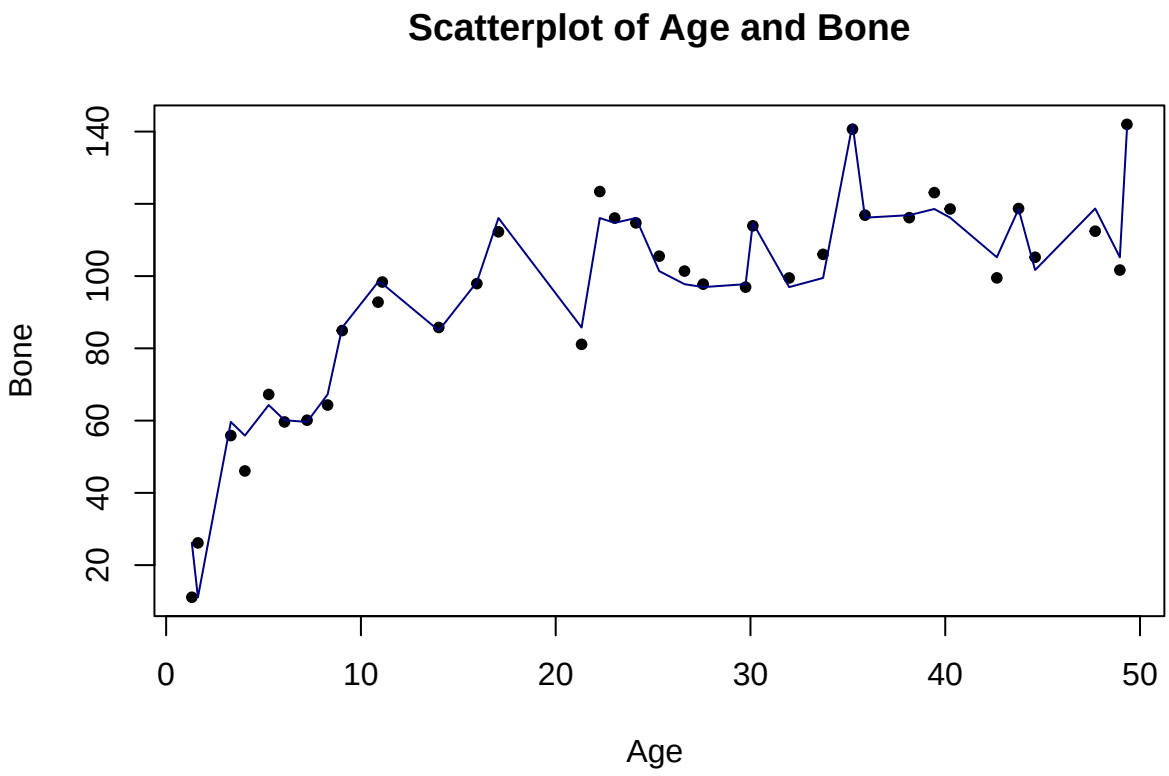
```
set.seed(123)
# Final optimal K via 10- fold CV
library("FNN")
SSEP <- function(yobs, yhat) sum((yobs-yhat)^2)

K <- 1:10
V <- 4
id.fold <- sample(1:V, size = NROW(D1), replace=T)
SSE <- rep(0, length(K))
for(k in 1:length(K)){
  for(v in 1:V){
    train1<- D1[id.fold!=v, ];
    train2<- D1[id.fold==v, ];
```

```
    yhat2 <- knn.reg(train=train1, y=train1$bone, test=train2, k=K[k], algorithm="kd_tree")$pred;
    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
```

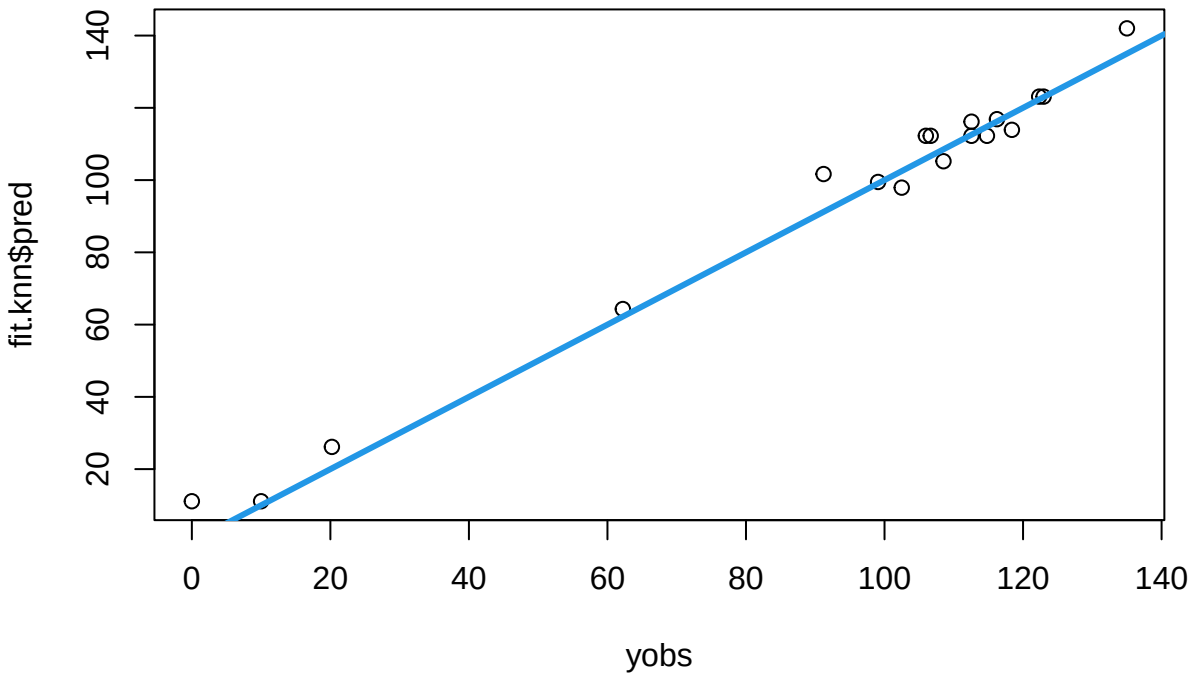
*We can observe from the output above that the SSE increases as k increases, thus, since k=1 has the lowest SSE, we chose k=1 as the optimal k*

```
# KNN regression
library("FNN")
fit.knn1 <- knn.reg(train=D1, y=D1$bone, k=1, algorithm="kd_tree");

plot(D1$age, D1$bone, main="Scatterplot of Age and Bone",
    xlab="Age ", ylab="Bone", pch=20)

lines(sort(D1$age), fit.knn1$pred[order(D1$age)],lty=1, col="navyblue")
```




*The above plot shows the fitted curve together with the scatter plot of the data using the optimal k*

```
# MEAN SQUARE ERROR FOR PREDICTION
fit.knn <- knn.reg(train=D1,test=D2,y=D1$bone, k=1, algorithm="kd_tree");
yobs <- D2[, 2]
plot(yobs,fit.knn$pred)
abline(a=0,b=1,col=4,lwd=3)
```




```
MSEP_knn <- mean((yobs-fit.knn$pred)^2)

MSEP_knn
```

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

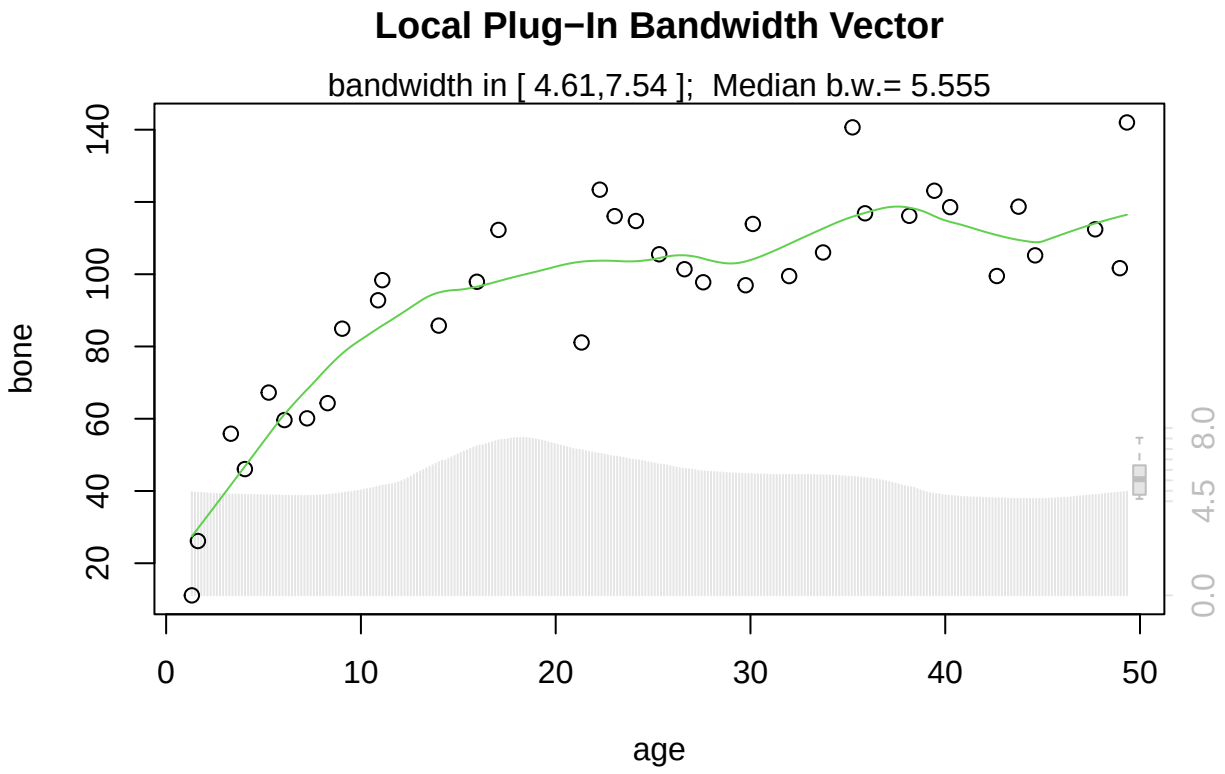
*When we apply KNN regression model with the optimal K, we have mean square error of prediction of 25.91247*

(4b) 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 and obtain the prediction MSE.

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

lofit <- lokerns(D1$age, D1$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
```

```
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 = D1, 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 our case, the bandwidth is within the interval 4.61 and 7.54*

```
# MEAN SQUARE ERROR FOR PREDICTION
Yhat_Pred_kernel <- predict(lofit, newdata = D2)
yobs <- D2[, 2]
#plot(yobs,Yhat_Pred_kernel)
#abline(a=0,b=1,col=4,lwd=3)
MSEP_kernel <- mean((yobs-Yhat_Pred_kernel$y)^2)
MSEP_kernel
```

```
## [1] 1619.397
```

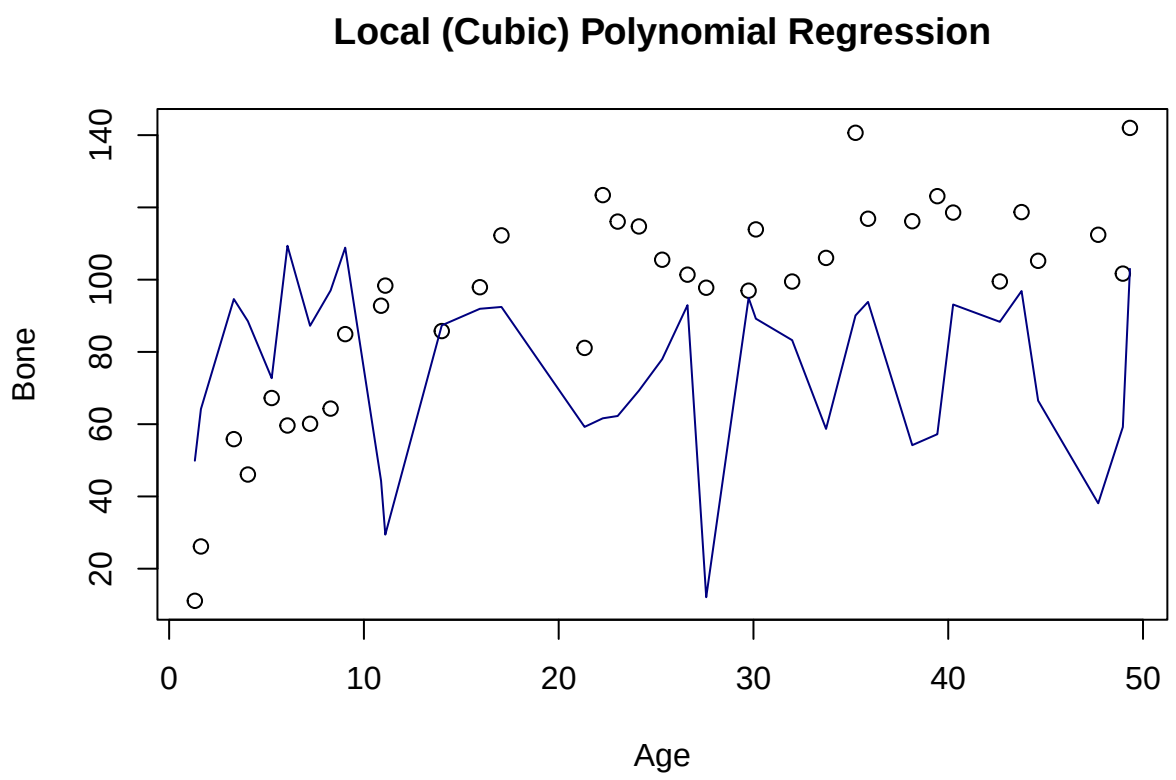
*We can observe a prediction error of 1619.397 from the above*

(4c) 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 and obtain the prediction MSE.

```
# Local (Cubic) Polynomial Regression
library(locpol)
fit.local <- locpol(bone~age, data=D1, deg=3, kernel=EpaK,bw =5)
Yhat_Pred_local <- locpol(bone~age, data=D1, xeval=D2$age, deg=3, kernel=EpaK,bw =6)$lpFit$bone
```

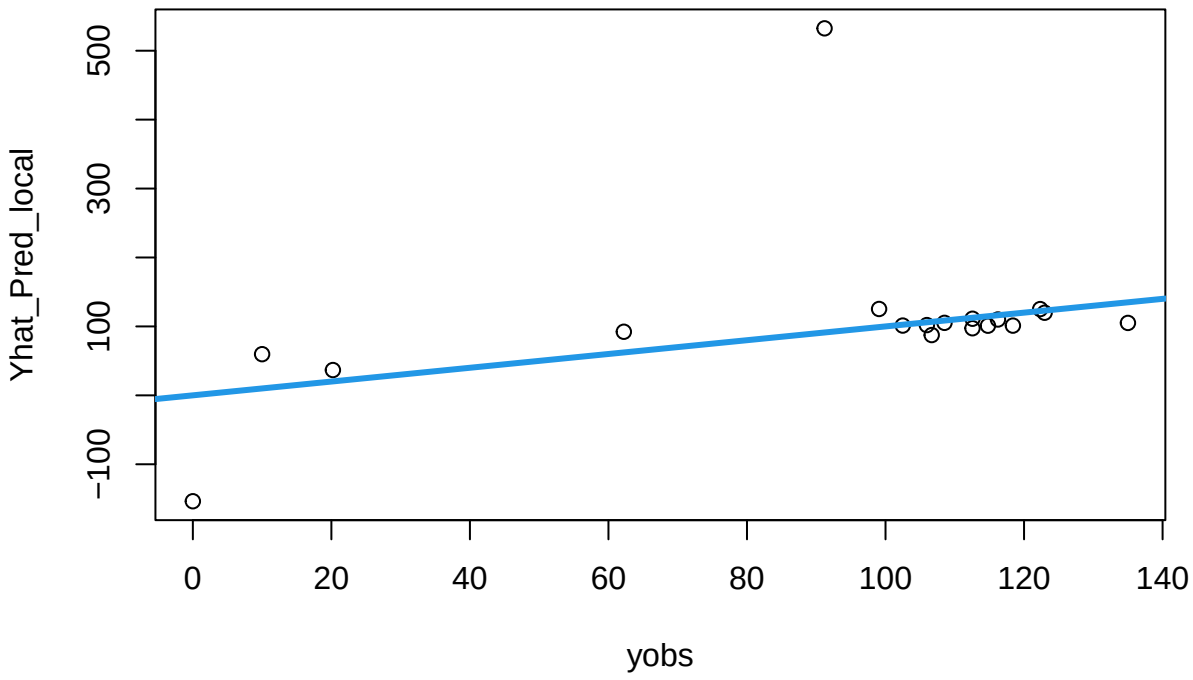
*Here, we chose a Smoothing parameter, bandwidth of 5*

```
plot(D1$age, D1$bone, xlab = "Age", ylab = "Bone", main="Local (Cubic) Polynomial Regression")
lines(sort(D1$age), fitted(fit.local)[order(D1$age)], lty=1, col="navyblue",cex=10)
```




*The above plot shows the local cubic polynomial regression. The plot seems interesting because it is unable to interpolate the data points*

```
yobs <- D2[, 2]
MSEP_local <- mean((yobs-Yhat_Pred_local)^2)
plot(yobs,Yhat_Pred_local)
abline(a=0,b=1,col=4,lwd=3)
```




```
MSEP_local
```

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

*We can observe MSEP of 12482.95 for the local polynomial regression which seems quite high*

(5a) 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.

```
library(splines)
attach(D1)
bs(D1$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
```

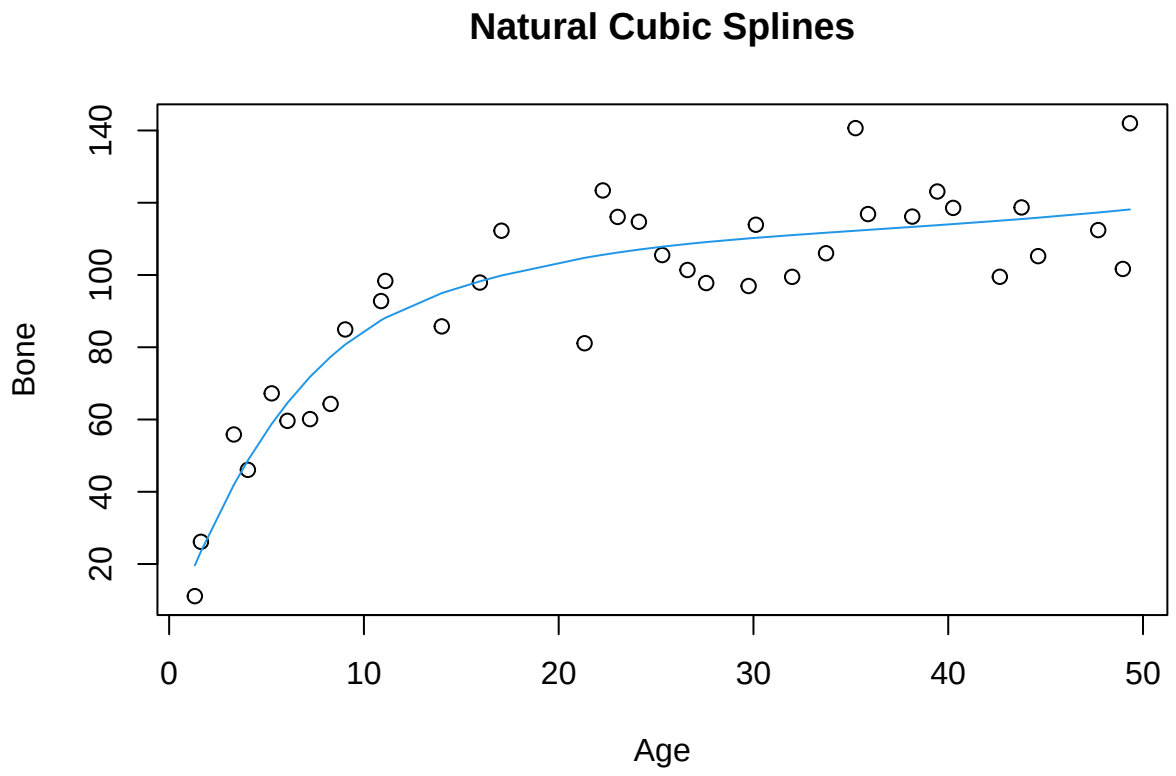
```
## [11,] 6.486711e-02 5.381942e-01 3.800569e-01 1.688171e-02 0.0000000000
## [12,] 5.147309e-02 5.169309e-01 4.085211e-01 2.307499e-02 0.0000000000
## [13,] 6.498477e-02 6.636947e-04 1.460960e-06 0.000000e+00 0.0000000000
## [14,] 0.000000e+00 3.824849e-03 9.669041e-02 5.281041e-01 0.3713806236
## [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"
```

```
spl1 <- lm(bone ~ bs(age, df = 5), data = D1)
summary(spl1 )
```

```
##
## Call:
## lm(formula = bone ~ bs(age, df = 5), data = D1)
##
## 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 = D1, xlab = "Age", ylab = "Bone", main = "Natural Cubic Splines")
spd <- seq(min(D1$age), max(D1$age), len = 20)
lines(sort(D1$age), spl1$fitted.values[order(D1$age)], lty=1, col=4)
```




*The above plot shows the resultant curve when we apply the natural regression splines to the training set D1*

```
Pred_spline <- predict(spl1,D2)
```

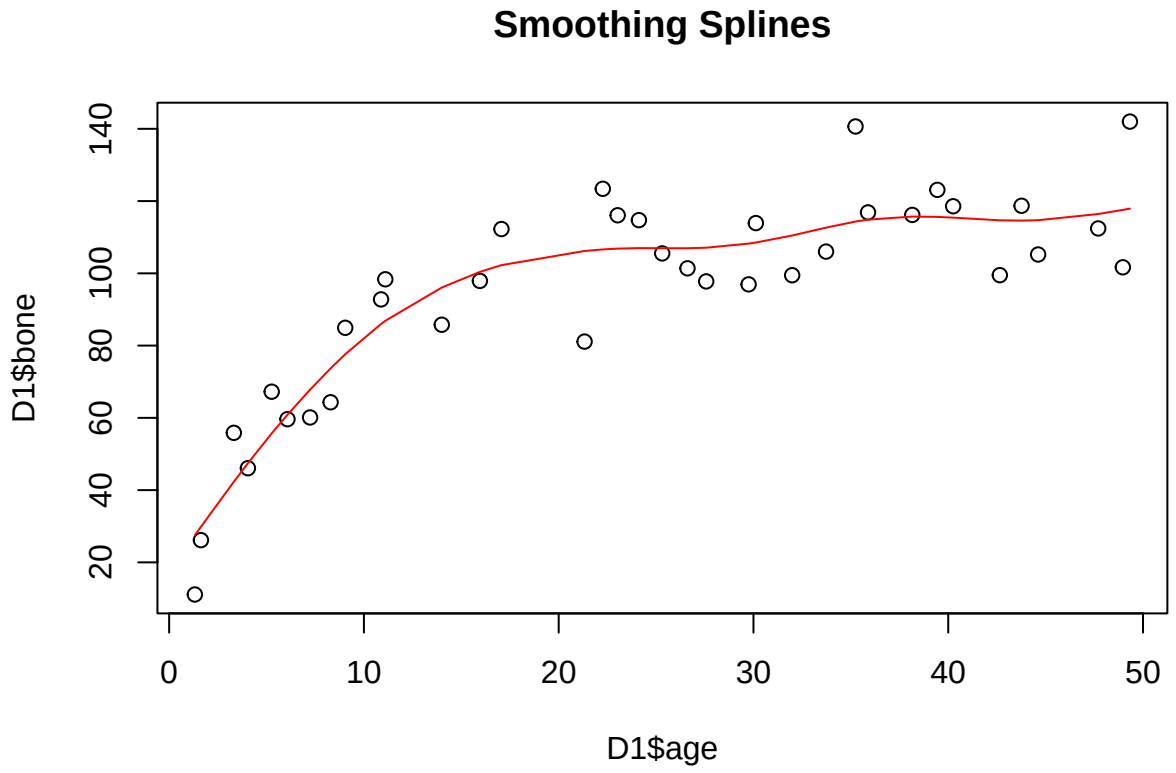
```
yobs <- D2[, 2]
MSEP_spline <- mean((yobs-Pred_spline)^2)
MSEP_spline
```

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

*We can observe MSEP of 263.8793 when we apply the natural cubic splines*

(5b) 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.

```
plot(D1$age, D1$bone, main = "Smoothing Splines")
tsmspl <- smooth.spline(D1$age, D1$bone)
lines(tsmspl, col = "red")
```




```
tsmspl
```

```
## Call:
## smooth.spline(x = D1$age, y = D1$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
```

*Here, i used generalized cross-validation (GCV) for the smoothing parameter estimation and a tuning parameter df = 5.75 was used.*

```
Pred_smooth <- predict(tsmspl,D2$age)
yobs <- D2[, 2]
MSEP_smooth <- mean((yobs-Pred_smooth$y)^2)
MSEP_smooth
```

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

*We can observe MSEP of 275.0591 for the smoothing splines*

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

```
Measure <- c(MSEP1, MSEP_knn,MSEP_kernel,MSEP_local,MSEP_spline,MSEP_smooth)
Measures <- data.frame("Method"= c("Asymptotic exponential model","KNN regression","Kernel regression",
knitr::kable(Measures, align = "lc")
```

| Method | Prediction.MSE.Measures |
|---|---|
| Asymptotic exponential model | 236.08228 |
| KNN regression | 25.91247 |
| Kernel regression | 1619.39713 |
| Local cubic polynomial | 12482.95403 |
| Natural cubic spline | 263.87930 |
| Smoothing Splines | 275.05908 |

*From the table above, we can observe that KNN regression gives the best result and the Local cubic polynomial gives the worst result*