C3M3_peer_review

June 26, 2023

1 C3M3: Peer Reviewed Assignment

1.0.1 Outline:

The objectives for this assignment:

1. Implement kernel smoothing in R and interpret the results.
2. Implement smoothing splines as an alternative to kernel estimation.
3. Implement and interpret the loess smoother in R.
4. Compare and contrast nonparametric smoothing methods.

General tips:

1. Read the questions carefully to understand what is being asked.
2. This work will be reviewed by another human, so make sure that you are clear and concise in what your explanations and answers.

[1]:
```
# Load Required Packages
library(ggplot2)
library(mgcv)
```

```
Loading required package: nlme

This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

2 Problem 1: Advertising data

The following dataset containts measurements related to the impact of three advertising medias on sales of a product, P. The variables are:

- `youtube`: the advertising budget allocated to YouTube. Measured in thousands of dollars;
- `facebook`: the advertising budget allocated to Facebook. Measured in thousands of dollars; and
- `newspaper`: the advertising budget allocated to a local newspaper. Measured in thousands of dollars.

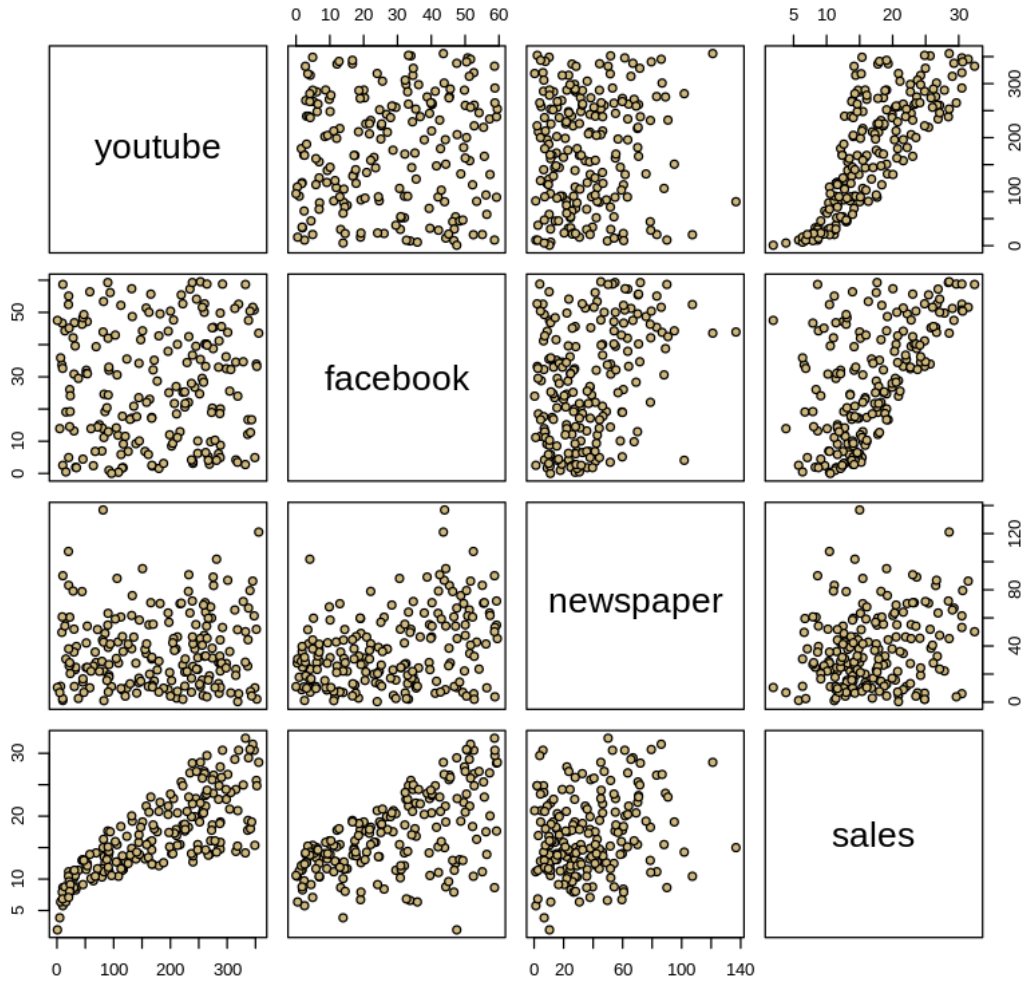
- `sales`: the value in the i^{th} row of the sales column is a measurement of the sales (in thousands of units) for product P for company i.

The advertising data treat "a company selling product P" as the statistical unit, and "all companies selling product P" as the population. We assume that the $n = 200$ companies in the dataset were chosen at random from the population (a strong assumption!).

First, we load the data, plot it, and split it into a training set (`train_marketing`) and a test set (`test_marketing`).

[2]:
```
# Load in the data
marketing = read.csv("marketing.txt", sep="")
summary(marketing)
pairs(marketing, main = "Marketing Data", pch = 21,
      bg = c("#CFB87C"))
```

```
    youtube          facebook       newspaper          sales      
 Min.   :  0.84   Min.   : 0.00   Min.   :  0.36   Min.   : 1.92  
 1st Qu.: 89.25   1st Qu.:11.97   1st Qu.: 15.30   1st Qu.:12.45  
 Median :179.70   Median :27.48   Median : 30.90   Median :15.48  
 Mean   :176.45   Mean   :27.92   Mean   : 36.66   Mean   :16.83  
 3rd Qu.:262.59   3rd Qu.:43.83   3rd Qu.: 54.12   3rd Qu.:20.88  
 Max.   :355.68   Max.   :59.52   Max.   :136.80   Max.   :32.40  
```



[5]:
```
set.seed(1771) #set the random number generator seed.
n = floor(0.8 * nrow(marketing)) #find the number corresponding to 80% of the␣
 ↪data
index = sample(seq_len(nrow(marketing)), size = n) #randomly sample indicies to␣
 ↪be included in the training set

train_marketing = marketing[index, ] #set the training set to be the randomly␣
 ↪sampled rows of the dataframe
test_marketing = marketing[-index, ] #set the testing set to be the remaining␣
 ↪rows
dim(test_marketing) #check the dimensions
dim(train_marketing) #check the dimensions
```

1. 40 2. 4

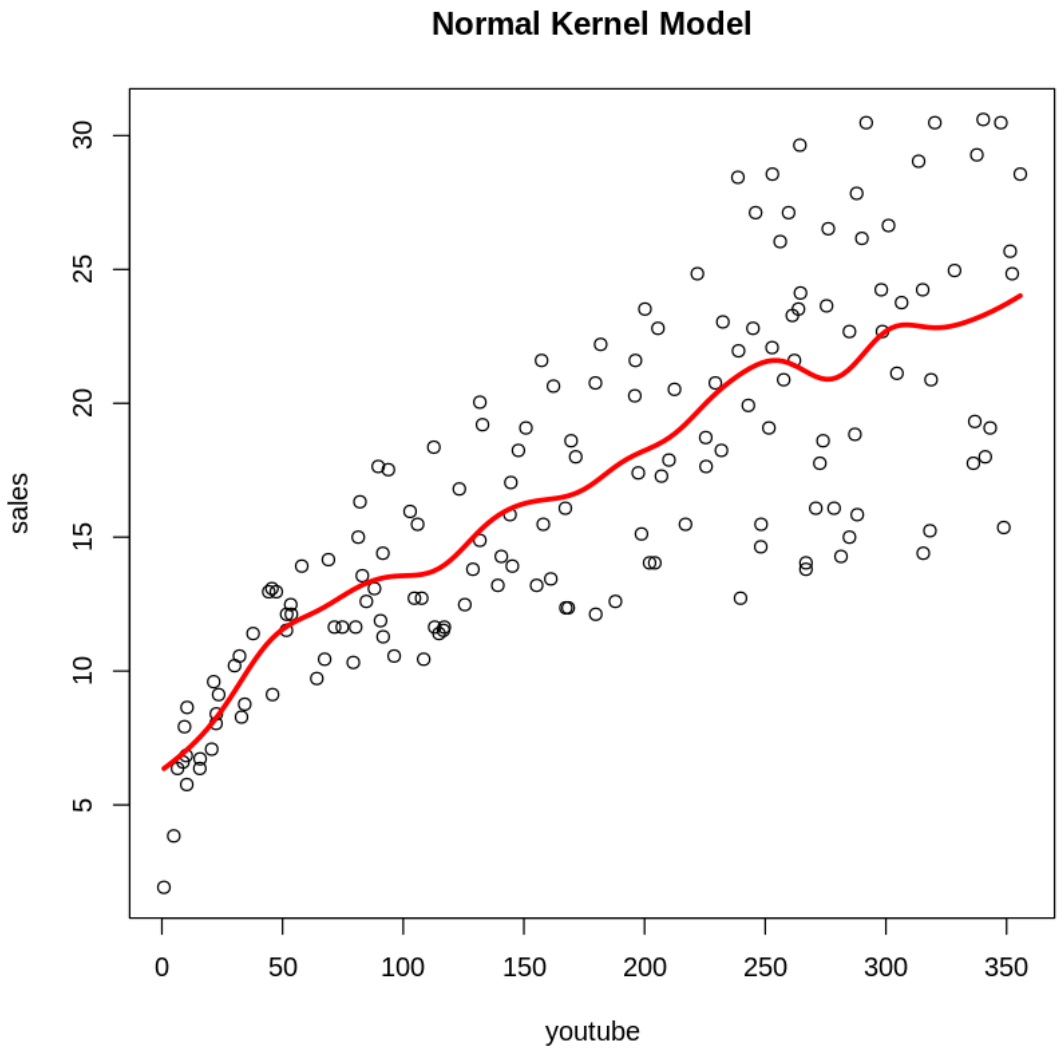
1. 160 2. 4

1.(a) Working with nonlinearity: Kernel regression

Note that the relationship between `sales` and `youtube` is nonlinear. This was a problem for us back in the first course in this specialization, when we modeled the data as if it were linear. For now, let's just focus on the relationship between `sales` and `youtube`, omitting the other variables (future lessons on generalized additive models will allow us to bring back other predictors).

Using the `train_marketing` set, plot `sales` (response) against `youtube` (predictor), and then fit and overlay a kernel regression. Experiment with the bandwidth parameter until the smooth looks appropriate, or comment why no bandwidth is ideal. Justify your answer.

[30]:
```
plot( sales ~ youtube, data = train_marketing, main = "Normal Kernel Model")
lines(ksmooth(train_marketing$youtube,train_marketing$sales, kernel = 'normal',␣
 ↪bandwidth = 35), col = 'red',lwd = 3)
```



The available fits struggle to capture the sharp nonlinearity effectively without introducing excessive variability in the upper sections of the YouTube data. For instance, with a bandwidth value of 10 and lower, the fit is able to track the steep nonlinearity but exhibits excessive fluctuations. On the other hand, with a bandwidth value of 35 and above, the fit becomes smoother in the higher regions of YouTube but fails to capture the sharp nonlinearity effectively.

1.(b) Working with nonlinearity: Smoothing spline regression

Again, using the `train_marketing` set, plot `sales` (response) against `youtube` (predictor). This time, fit and overlay a smoothing spline regression model. Experiment with the smoothing parameter until the smooth looks appropriate. Explain why it's appropriate and justify your answer.

[29]:
```
plot(sales ~ youtube, data = train_marketing, main = "Spline regression model")
```

```
lines(smooth.spline(train_marketing$youtube,train_marketing$sales, spar = 0.7),␣
↪col = "red", lwd = 3)
```

This approach demonstrates improved performance in capturing the steep nonlinearity while avoiding excessive variability in the upper regions of YouTube. By setting the smoothing parameter (spar) to 0.7, we are able to track the sharp nonlinearity without introducing excessive "wiggles" in the fit. Although the result is not perfect, it shows promise in achieving a more balanced representation of the data.

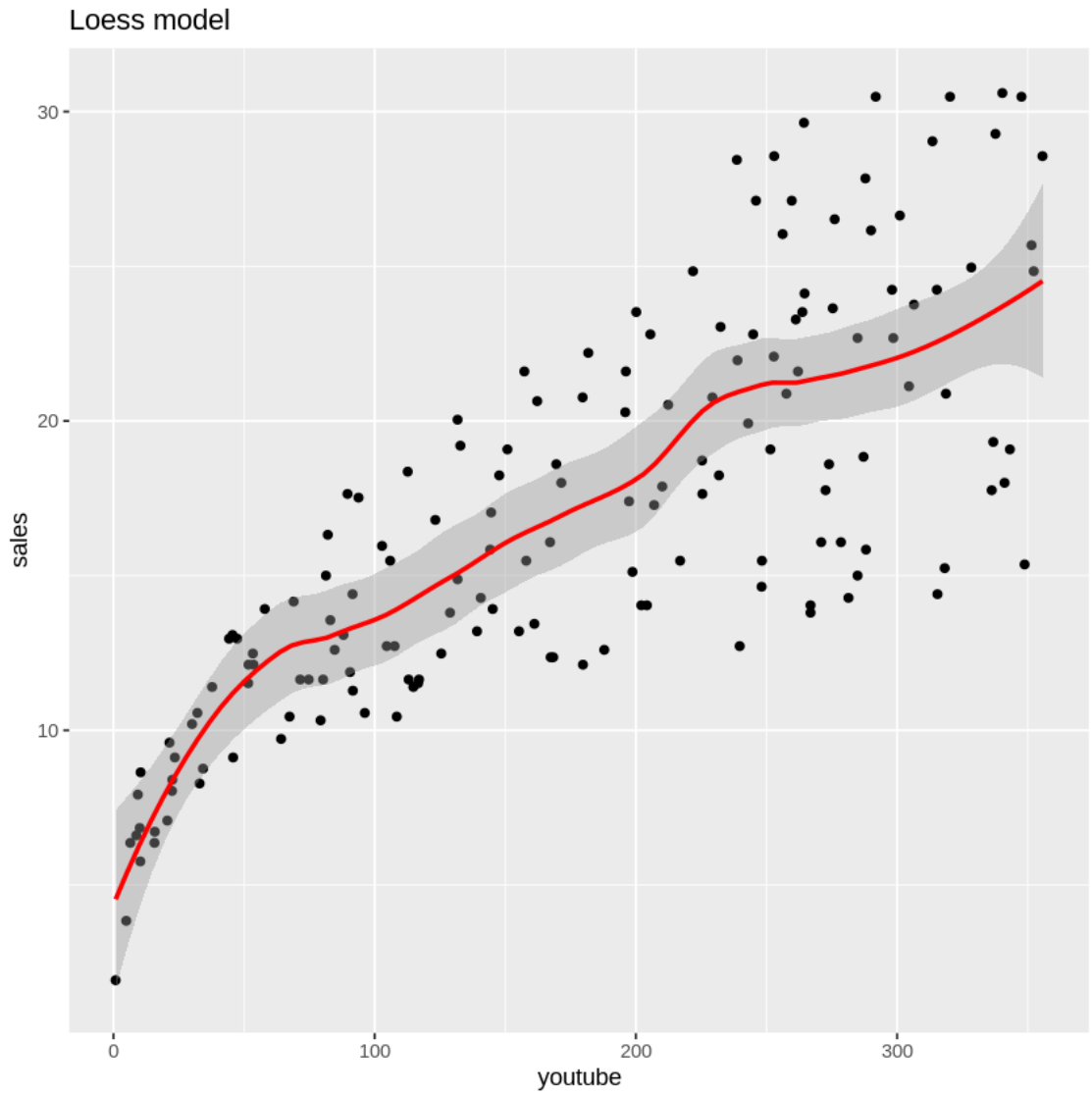
1.(c) Working with nonlinearity: Loess

Again, using the `train_marketing` set, plot `sales` (response) against `youtube` (predictor). This time, fit and overlay a loess regression model. You can use the `loess()` function in a similar way as the `lm()` function. Experiment with the smoothing parameter (`span` in the `geom_smooth()`

function) until the smooth looks appropriate. Explain why it's appropriate and justify your answer.

[98]:
```
lomod = loess(sales ~ youtube, data = train_marketing, span = 0.4)
ggplot(train_marketing, aes(x = youtube, y = sales))+
    geom_point() +
    geom_smooth(method = 'loess',color = 'red', lwd = 1, span = 0.4) +
    labs(title = "Loess model")
```

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



Utilizing a span value of 0.4 yields a satisfactory fit. It effectively captures the nonlinearity observed in the lower values of YouTube and avoids excessive fluctuations in the upper regions. This choice

of span strikes a good balance, resulting in a fit that reasonably accounts for the nonlinear patterns in the data.

1.(d) A prediction metric

Compare the models using the mean squared prediction error (MSPE) on the `test_marketing` dataset. That is, calculate the MSPE for your kernel regression, smoothing spline regression, and loess model, and identify which model is best in terms of this metric.

Remember, the MSPE is given by

$$MSPE = \frac{1}{k}\sum_{i=1}^{k}(y_i^\star - \widehat{y}_i^\star)^2$$

where $y_i^\star$ are the observed response values in the test set and $\widehat{y}_i^\star$ are the predicted values for the test set (using the model fit on the training set).

[99]:
```
mod_kr = ksmooth(train_marketing$youtube, train_marketing$sales, kernel =␣
↪"normal", 35, x.points = test_marketing$youtube)
cat("The MSPE for kernel regression is", mean((test_marketing$sales -␣
↪mod_kr$y)^2),".")

mod_ss = predict(smooth.spline(train_marketing$youtube,train_marketing$sales,␣
↪spar = 0.7), x = test_marketing$youtube);
cat("The MSPE for smoothing spline is",mean((test_marketing$sales -␣
↪mod_ss$y)^2), ".")
mod_loess = predict(lomod, newdata = test_marketing$youtube);
cat("The MSPE for loess is", mean((test_marketing$sales - mod_loess)^2), ".")
```

```
The MSPE for kernel regression is 64.81 .The MSPE for smoothing spline is
18.1893 .The MSPE for loess is 18.05551 .
```

Based on the data analysis performed, it can be concluded that the loess regression model exhibits the smallest Mean Squared Prediction Error (MSPE) among the evaluated models. This indicates that the loess regression provides the best predictive performance for these particular data.

3 Problem 2: Simulations!

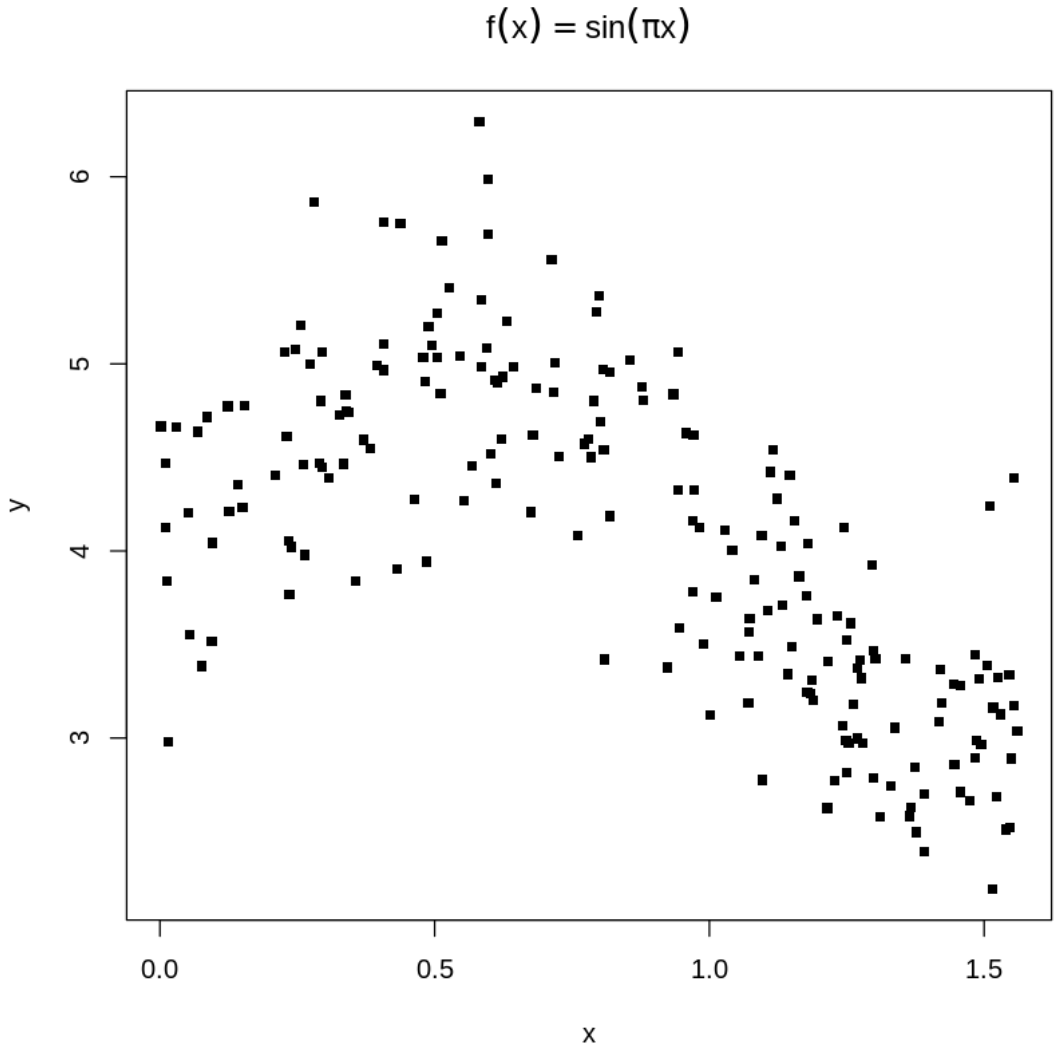
Simulate data (one predictor and one response) with your own nonlinear relationship. Provide an explanation of how you generated the data. Then answer the questions above (1.(a) - 1.(d)) using your simulated data.

[113]:
```
#simulated data
set.seed(12345)
library(ggplot2)
n = 200
```

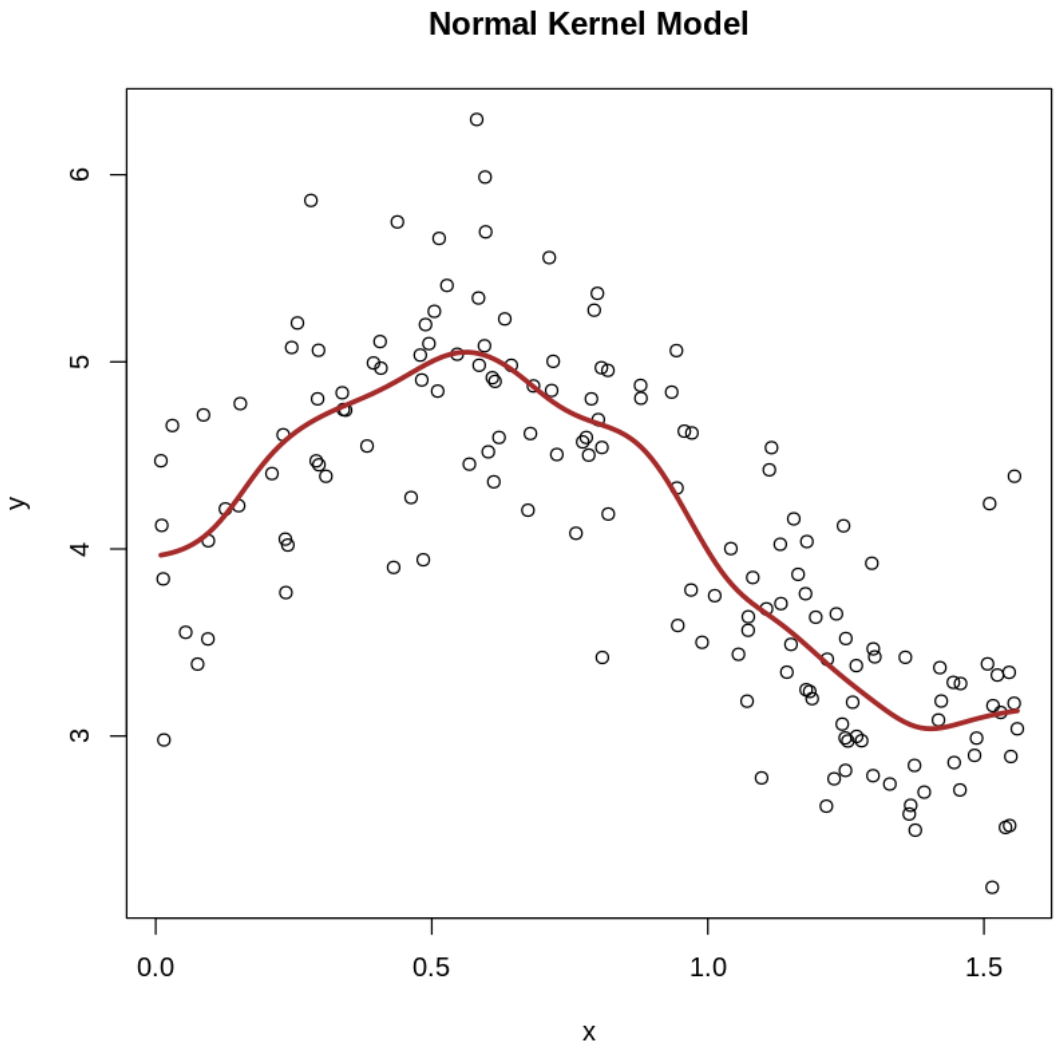
```
x = runif(n, 0, pi/2)
y = sin(pi*x) + rnorm(n, 0, 0.5) + 4
d = data.frame(x=x,y=y)

set.seed(1555)
n = floor(0.8 * nrow(d))
index = sample(seq_len(nrow(d)), size = n)

train = d[index, ]
test = d[-index, ]
dim(train)
dim(test)



plot(y ~ x, main = expression(f(x) == sin(pi*x)), pch = 15, cex=0.8)
```

1. 160 2. 2

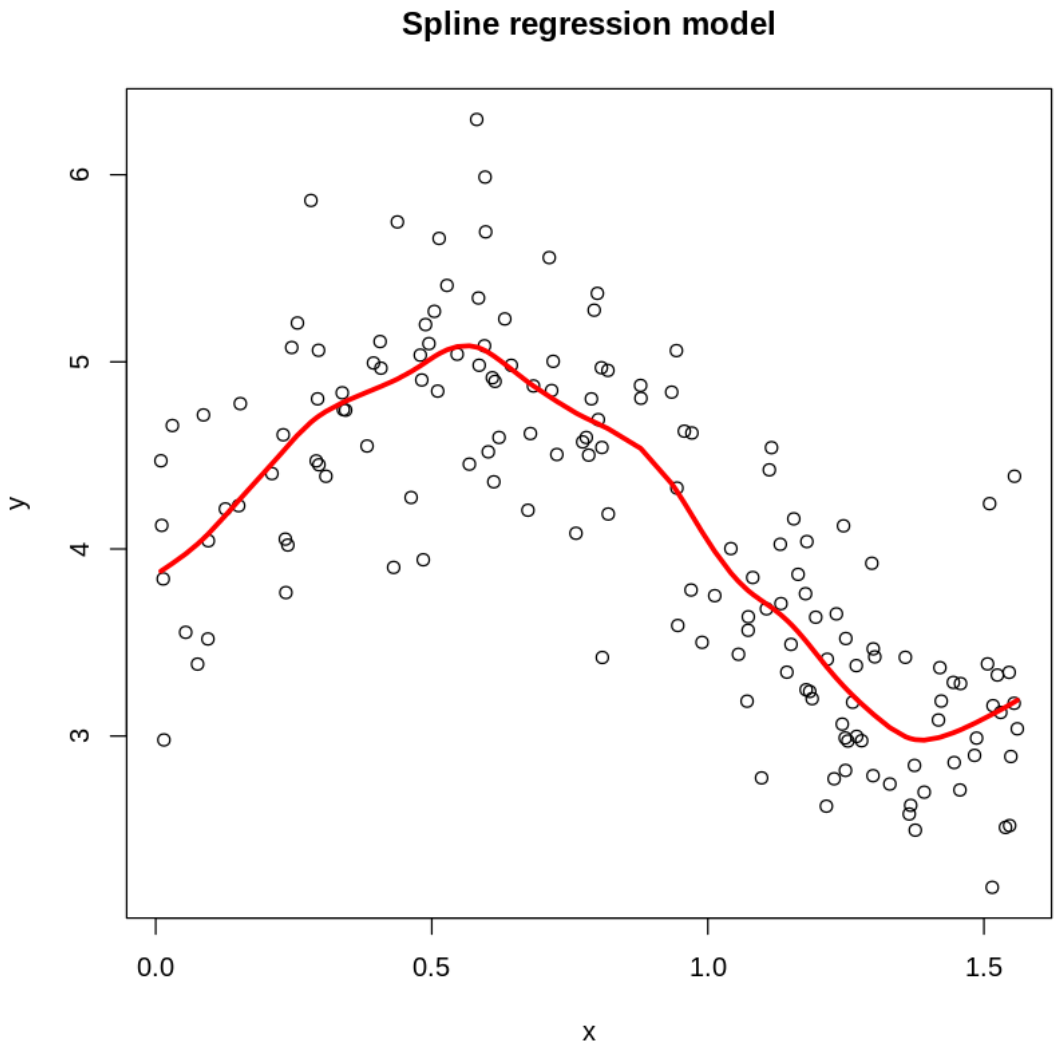
1. 40 2. 2



[70]:
```
#1.a
plot( y ~ x, data = train, main = "Normal Kernel Model")
lines(ksmooth(train$x,train$y, kernel = 'normal', bandwidth = 0.2), col =␣
↪'brown',lwd = 3)
```

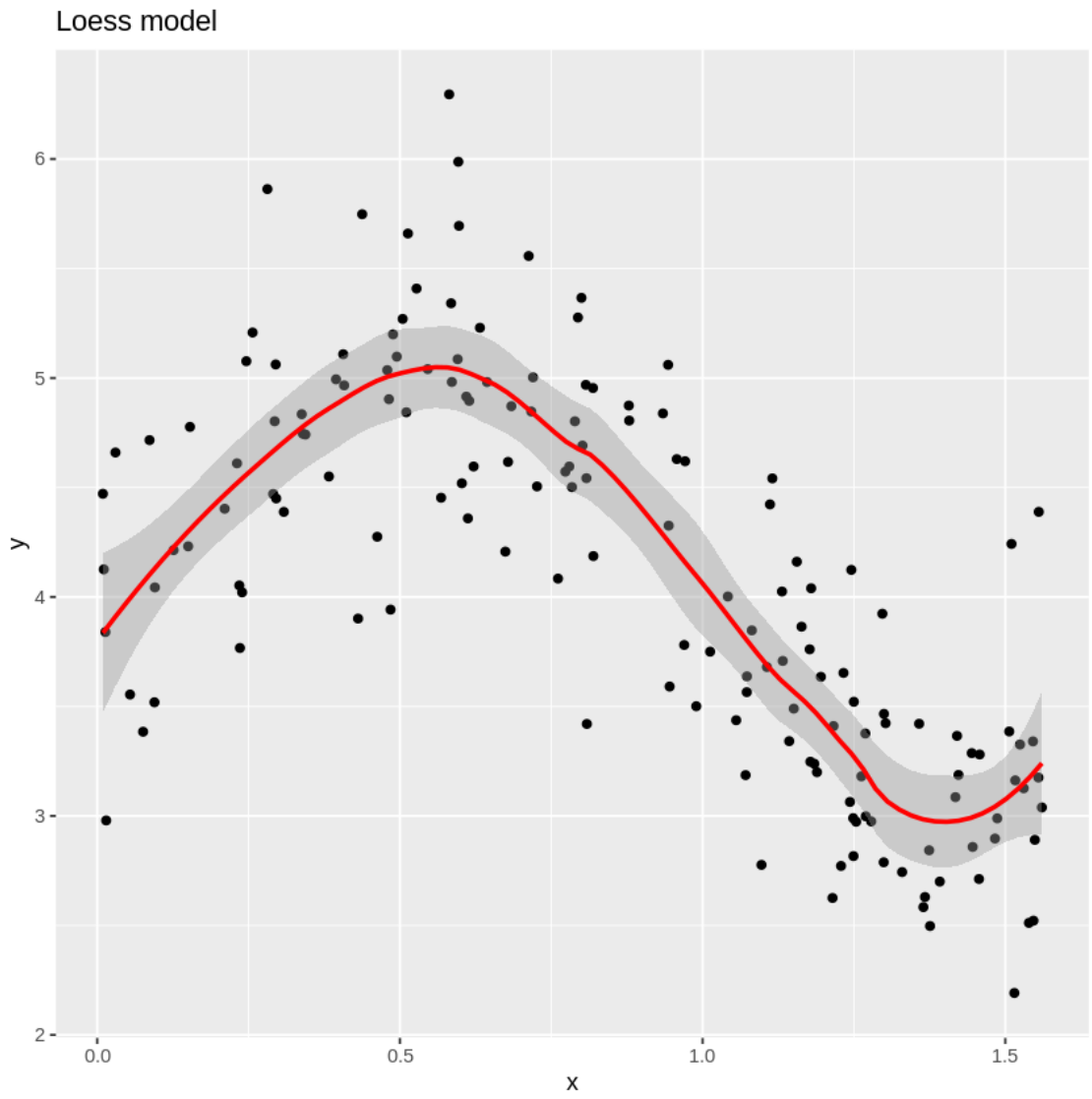


[80]:
```
#1.b
plot(y ~ x, data = train, main = "Spline regression model")
lines(smooth.spline(train$x,train$y, spar = 0.85), col = "red", lwd = 3)
```



[127]:
```
#1.c
lomod_2 = loess(y ~ x, data = train, span = 0.4)
ggplot(train, aes(x = x, y = y))+
    geom_point() +
    geom_smooth(method = 'loess',color = 'red', lwd = 1, span = 0.4) +
    labs(title = "Loess model")
```

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



[138]:
```
#1.d
kr = ksmooth(train$x, train$y, kernel = "normal", 35, x.points = test$x)
cat("The MSPE for kernel regression is", mean((test$y - kr$y)^2),".")

ss = predict(smooth.spline(train$x,train$y, spar = 0.7), x = test_sim$x);
cat("The MSPE for smoothing spline regression is",mean((test$y - ss$y)^2), ".")

#for some reason during predict operation we are getittin 1 NA, which is hard␣
 ↪to explain
#sum(is.na(loess_2))

#problematic_indices <- which(is.na(loess_2))
```

```
#test_clean <- test[-problematic_indices, ]


loess_2 = predict(lomod_2, newdata = test_clean$x);
cat("The MSPE for loess is", mean((test_clean$y - loess_2)^2), ".")
```

```
The MSPE for kernel regression is 0.6381185 .The MSPE for smoothing spline
regression is 0.2470695 .The MSPE for loess is 0.2239566 .
```

Based on the evaluation of the models, it can be concluded that the loess fit performs the best among the alternatives considered.

[ ]: