In [ ]:

In [ ]:

In [ ]:

In [ ]:

C3M3: Peer Reviewed Assignment

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.

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.

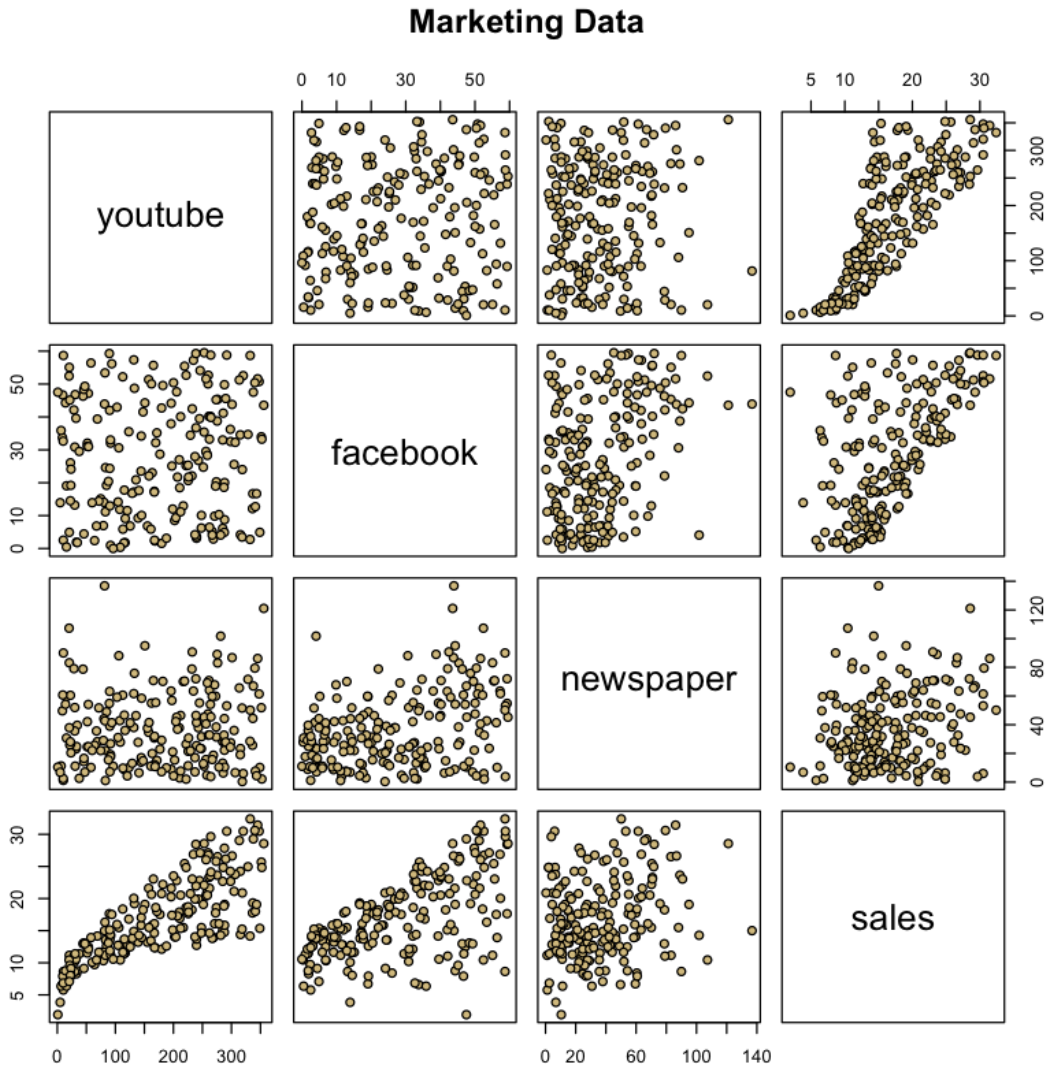
The advertising data treat "a company selling product P" as the statistical unit, and "all companies selling product P" as the population. We assume that the $n = 200$ companies in the dataset were chosen at random from the population (a strong assumption!).

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

In [1]:
```
library(RCurl) #a package that includes the function getURL(), whic
h allows for reading data from github.
library(ggplot2)
library(mgcv)

url = getURL("https://raw.githubusercontent.com/bzaharatos/-Statist
ical-Modeling-for-Data-Science-Applications/master/Modern%20Regress
ion%20Analysis%20/Datasets/marketing.txt")
marketing = read.csv(text = url, sep = "")
summary(marketing)
pairs(marketing, main = "Marketing Data", pch = 21,
      bg = c("#CFB87C"))
```

```
Loading required package: bitops
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Loading required package: nlme
This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

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



In [2]:
```
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 th
e 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
```

```
40  4
```

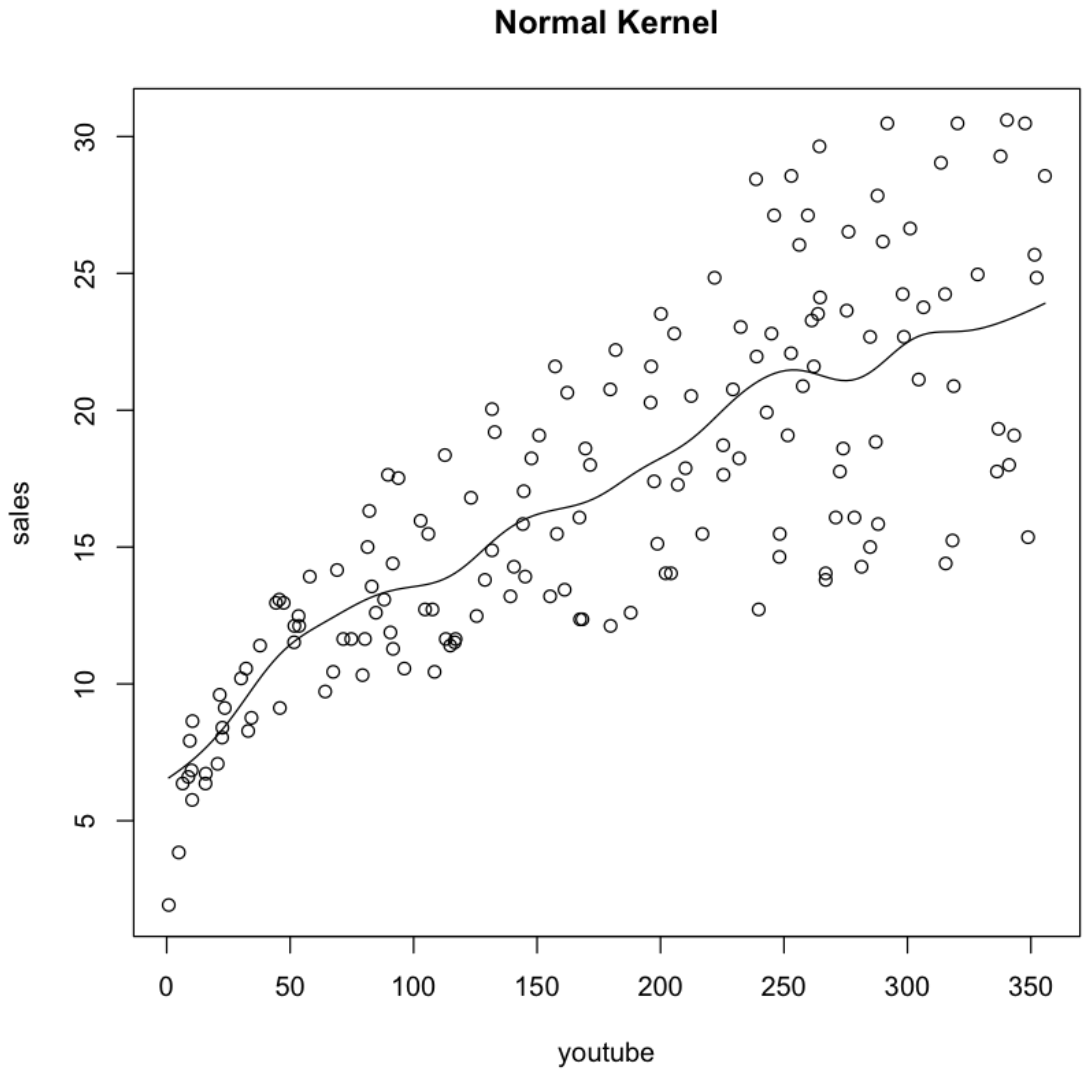
```
160  4
```

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.

In [3]:
```
plot(sales ~ youtube, data = train_marketing, main = "Normal Kernel")
lines(ksmooth(train_marketing$youtube, train_marketing$sales, kernel = "normal", 40))
```

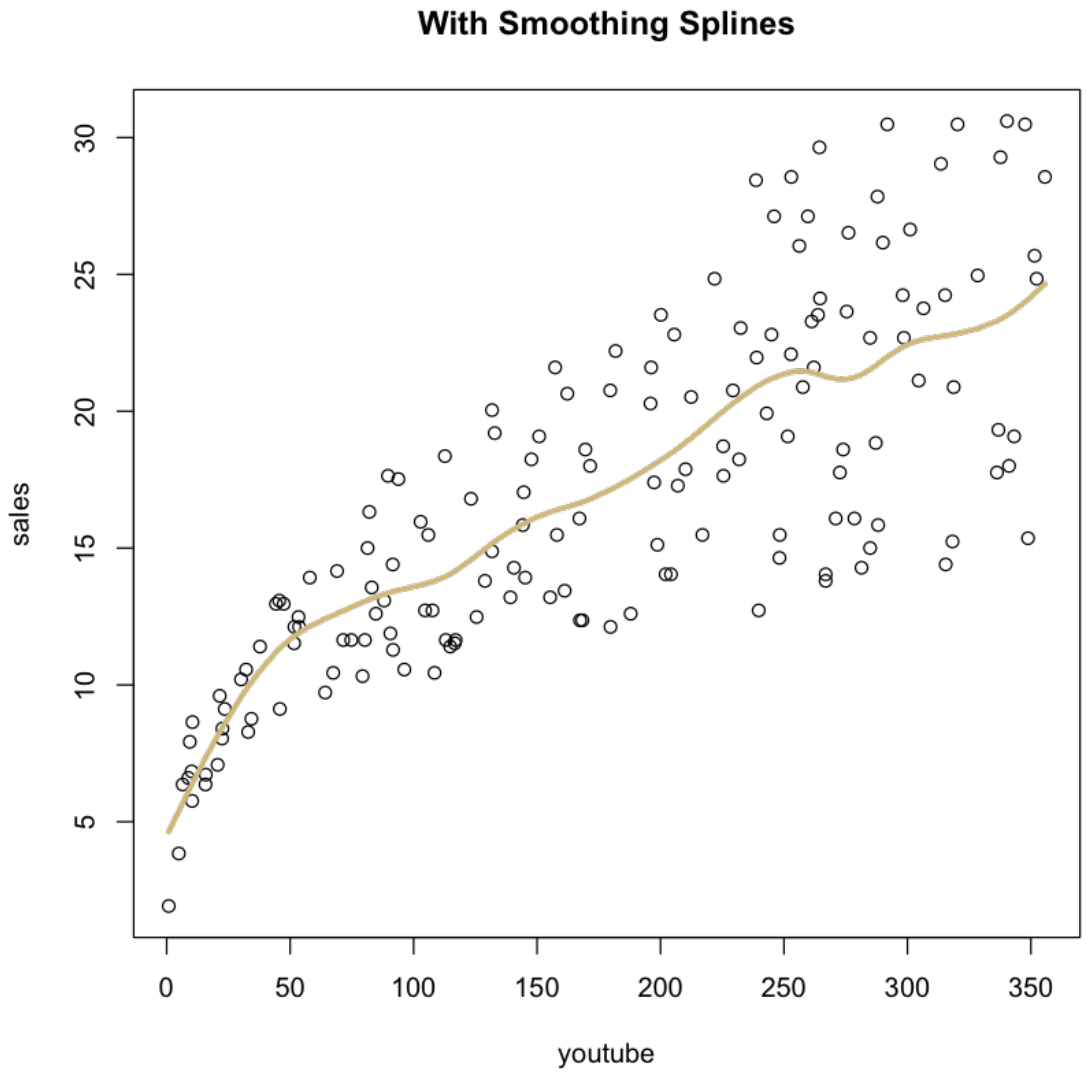


None of the fits are great at picking up the steep nonlinearity without being too "wiggly" in the upper regions of `youtube`. For example, `bandwidth = 10` tracks the steep nonlinearity but is too wiggly. `bandwidth = 40` misses the steep nonlinearity but is smoother in the higher regions of `youtube`.

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.

In [4]:
```
marketing.smooth = with(train_marketing, smooth.spline(y = sales, x
= youtube, spar = 0.75))

plot(sales ~ youtube, data = train_marketing, main = "With Smoothin
g Splines")
lines(marketing.smooth, col = "#CFB87C", lwd=3)
```



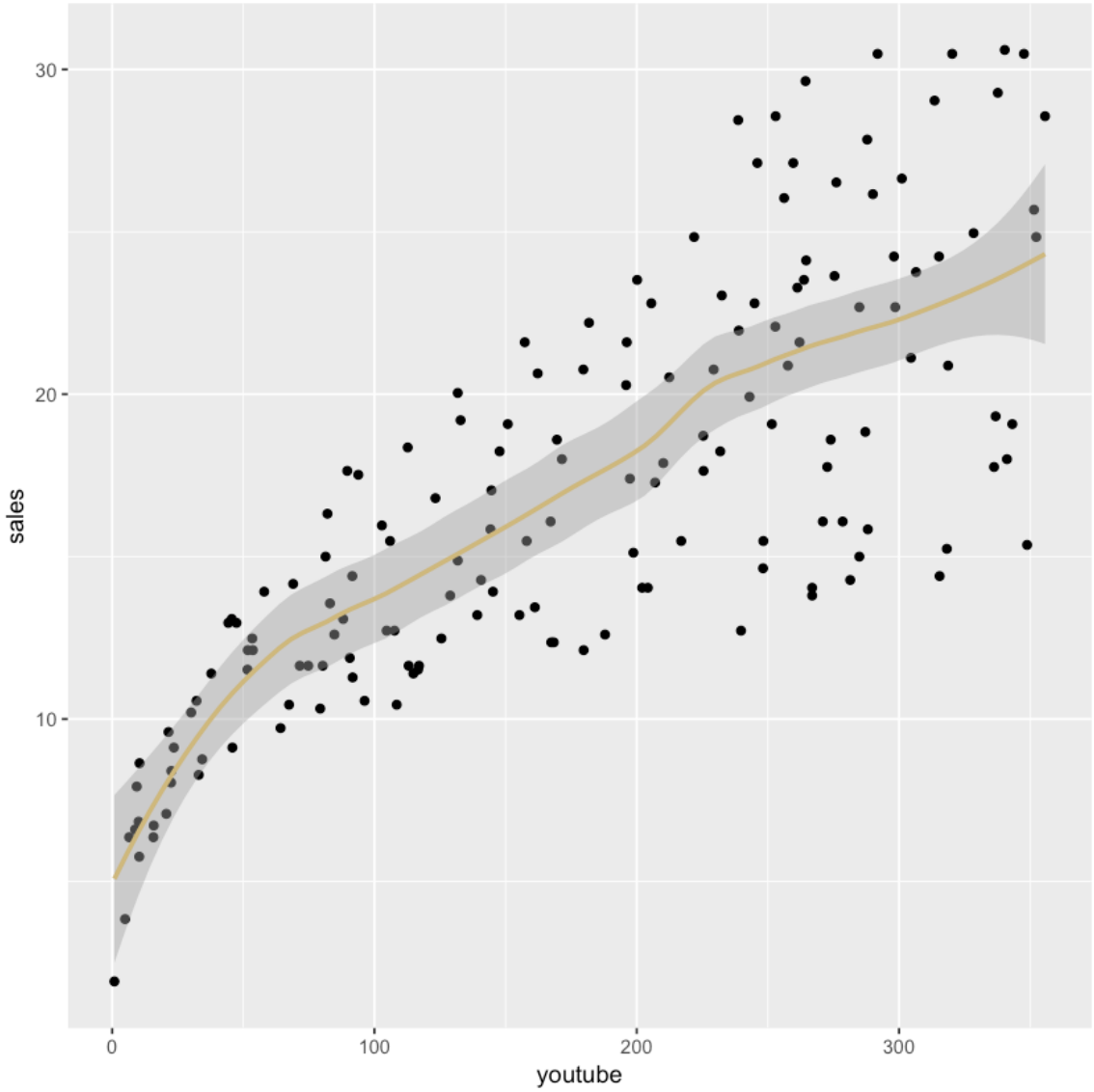
This method does a bit better at picking up the steep nonlinearity without being too "wiggly" in the upper regions of `youtube`. `spar = 0.75` tracks the steep nonlinearity and is not too wiggly (though not perfect).

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.

In [5]:
```
lr = loess(sales ~ youtube, train_marketing, span = 0.5) #loess fit to be used later

ggplot(train_marketing, aes(x = youtube, y = sales))+
    geom_point() +
    geom_smooth(color = "#CFB87C", span = 0.5)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The default `span = 0.5` provides a pretty nice fit! The nonlinearity at low values of `youtube` are reasonably well accounted for, and the fit in upper regions of `youtube` is not too wiggly.

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

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

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

In [6]:
```
yhat_kr = ksmooth(train_marketing$youtube, train_marketing$sales, k
ernel = "normal", 40, x.points = test_marketing$youtube)
cat("The MSPE for kernel regression is", mean((test_marketing$sales
- yhat_kr$y)^2),".")

yhat_ss = predict(marketing.smooth, x = test_marketing$youtube);
cat("The MSPE for smoothing spline regression is",mean((test_market
ing$sales - yhat_ss$y)^2), ".")
yhat_loess = predict(lr, newdata = test_marketing$youtube);
cat("The MSPE for loess regression is", mean((test_marketing$sales
- yhat_loess)^2), ".")
```

```
The MSPE for kernel regression is 64.45882 .The MSPE for smoothing s
pline regression is 18.12119 .The MSPE for loess regression is 18.11
483 .
```

The loess regression has the smallest MSPE for these data.

Problem 2: Simulations!

Simulate data (one predictor and one response) with your own nonlinear relationship. Then answer the questions above (1.(a) - 1.(d)) using your simulated data.

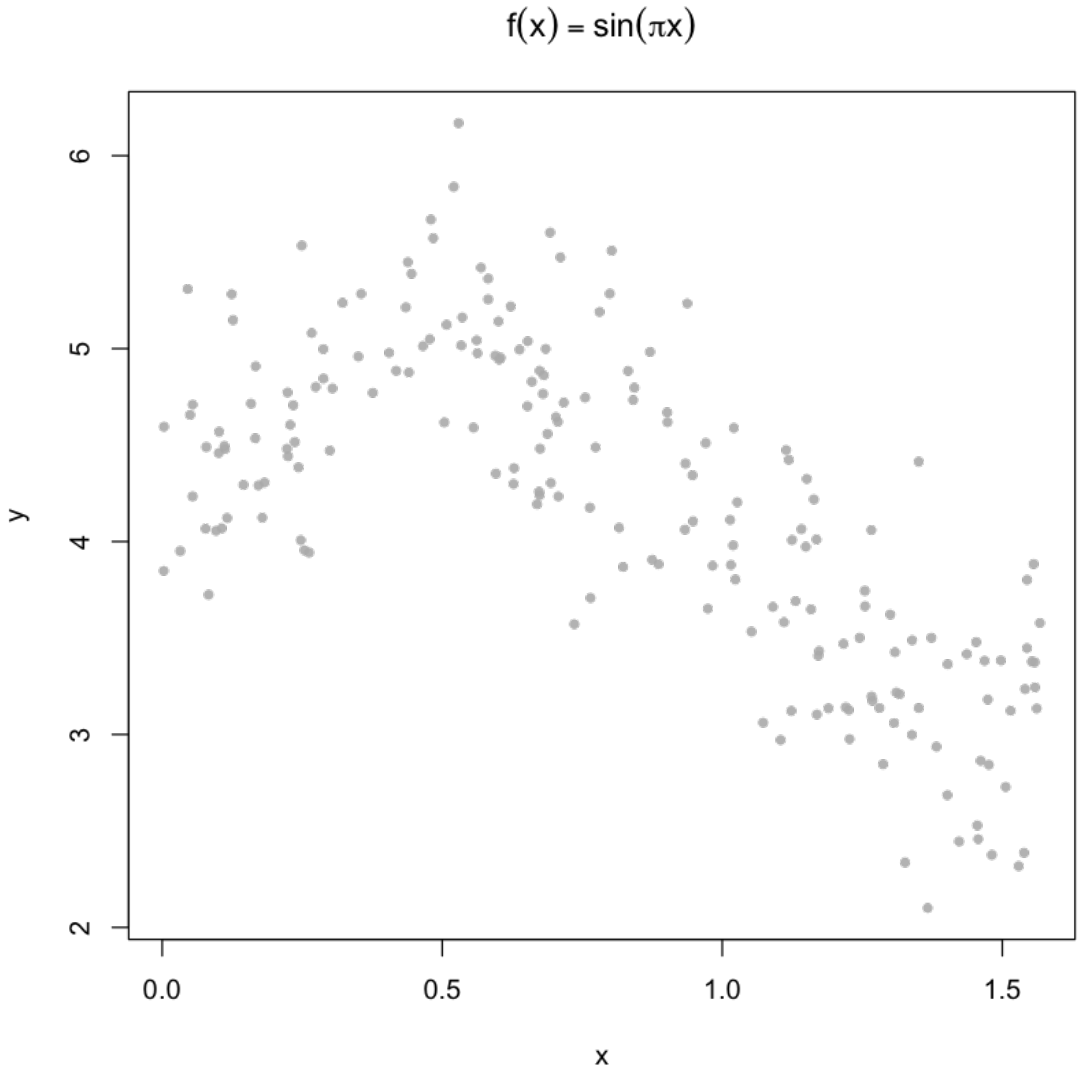
Simulate $n = 200$ (x, y) pairs in the following way: let x be random uniform numbers between zero and $\pi/2$. Let $y_i = \sin(\pi x_i) = \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} N(0, 0.5^2)$. Plot y as a function of x. Would a linear parametric model do well in explanation/prediction for this dataset?

In [7]:

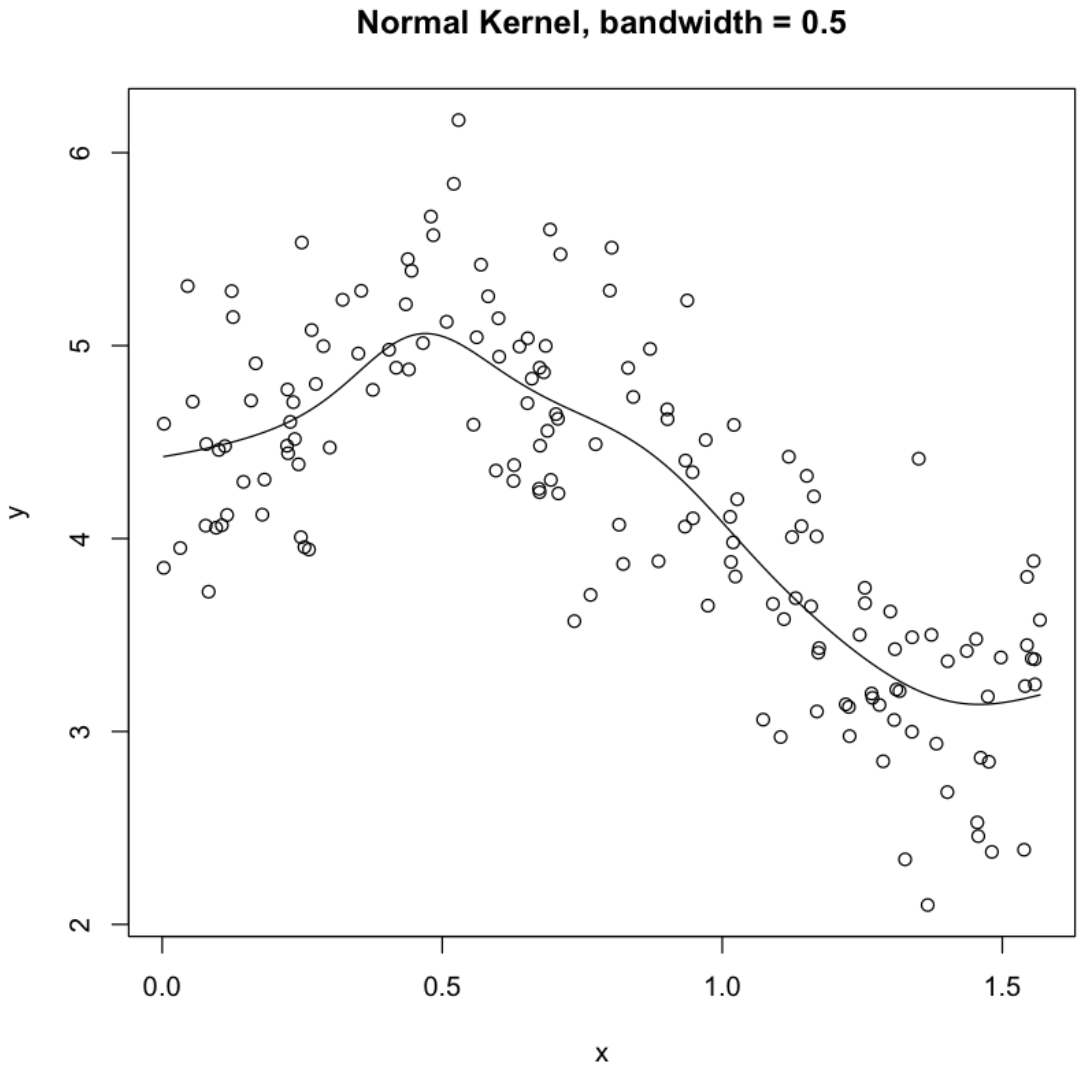
```
set.seed(88888)
library(ggplot2)
n = 200
x = runif(n, 0, pi/2)
y = sin(pi*x) + rnorm(n, 0, 0.5) + 4
d = data.frame(x=x,y=y)

set.seed(1771) #set the random number generator seed.
n = floor(0.8 * nrow(d)) #find the number corresponding to 80% of t
he data
index = sample(seq_len(nrow(d)), size = n) #randomly sample indicie
s to be included in the training set

train_sim = d[index, ] #set the training set to be the randomly sam
pled rows of the dataframe
test_sim = d[-index, ] #set the testing set to be the remaining row
s
dim(test_sim) #check the dimensions
dim(train_sim) #check the dimensions


plot(y ~ x, main = expression(f(x) == sin(pi*x)), pch = 16, cex=0.
8, col = alpha("darkgrey", 0.9))
```

40 2

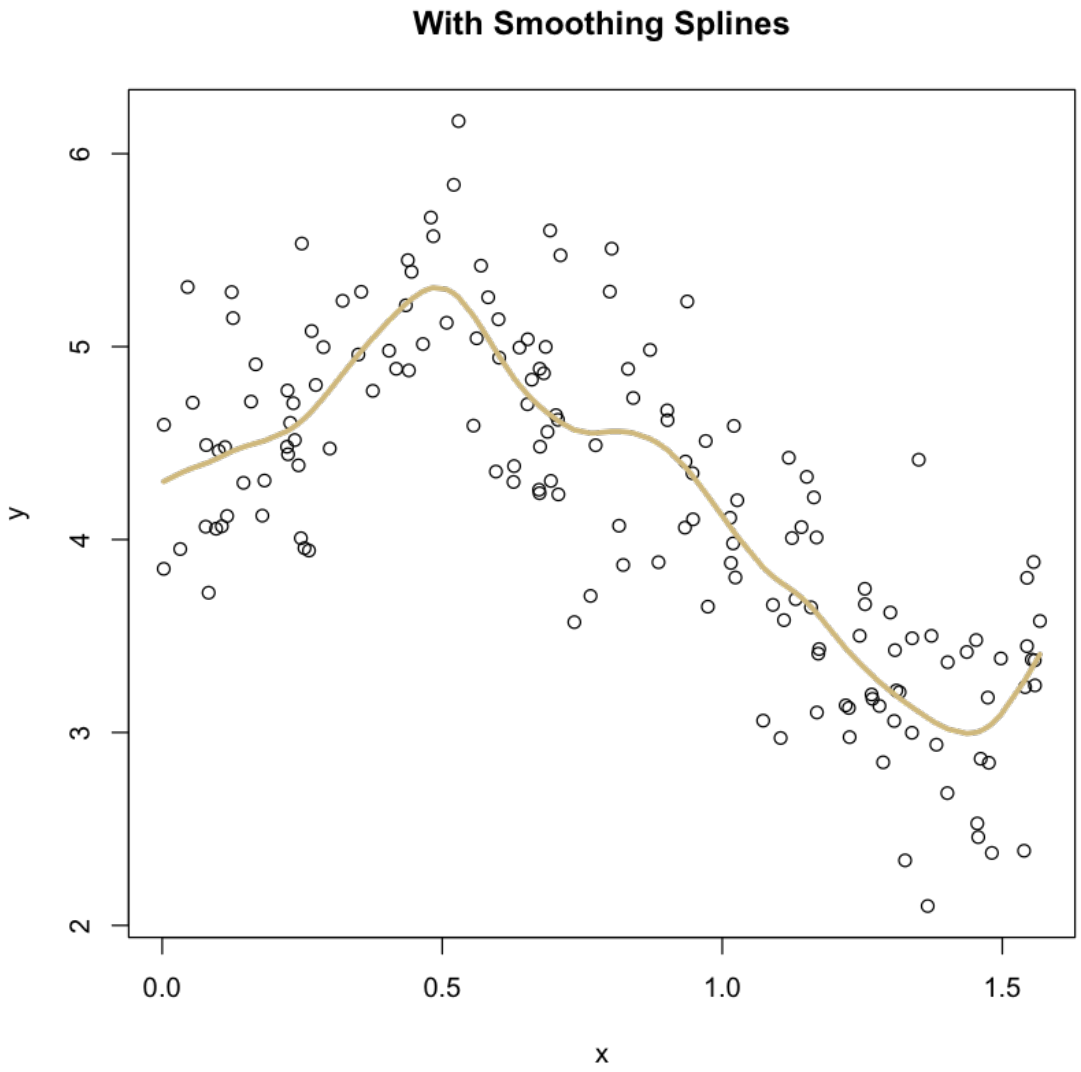
160 2



In [8]:
```
#1.a

plot(y ~ x, data = train_sim, main = "Normal Kernel, bandwidth = 0.
5")
lines(ksmooth(train_sim$x, train_sim$y, kernel = "normal", 0.3))
```



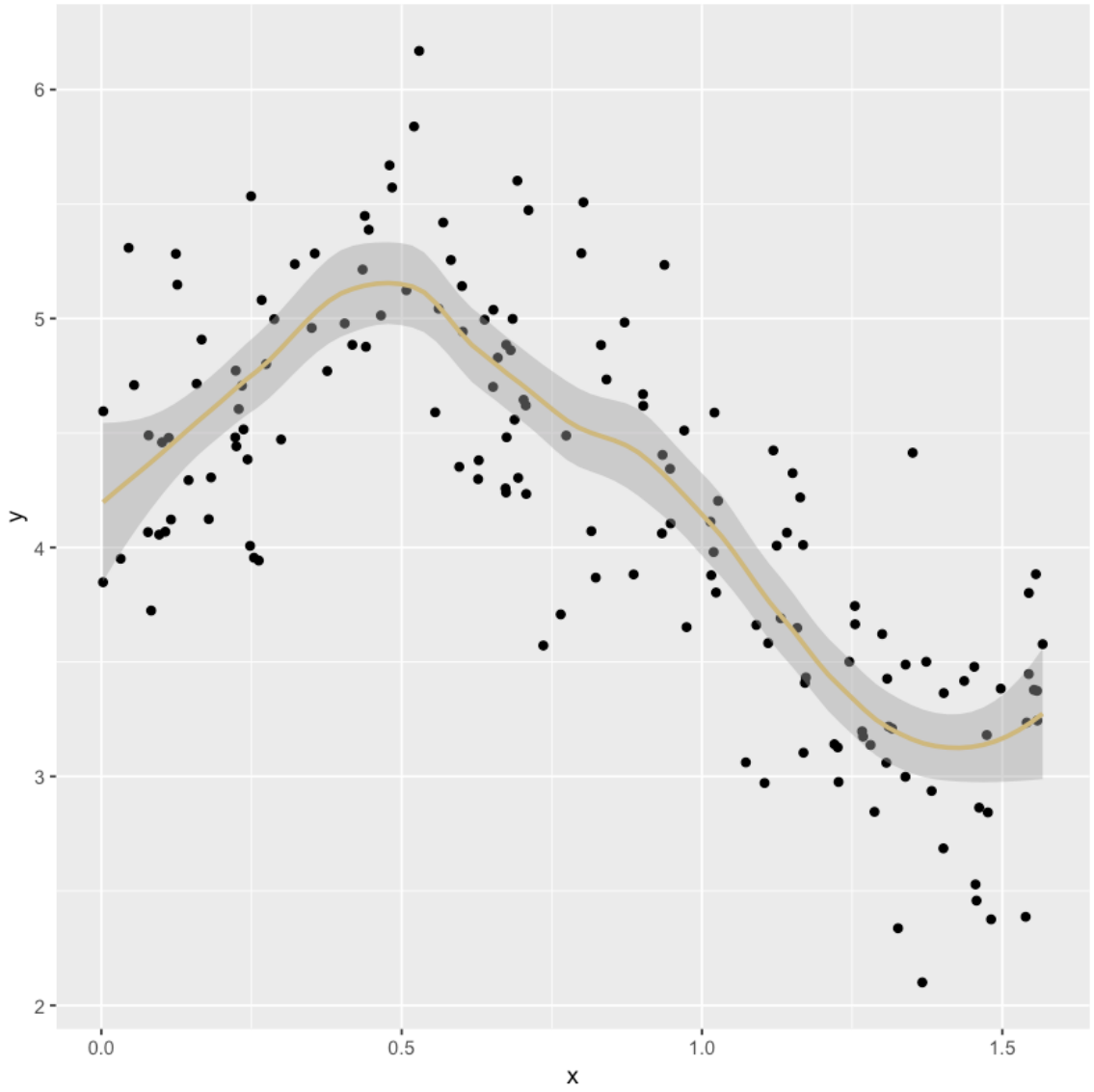
In [9]:
```
#1.b
sim.smooth = with(train_sim, smooth.spline(y = y, x = x, spar = 0.7
5))

plot(y ~ x, data = train_sim, main = "With Smoothing Splines")
lines(sim.smooth, col = "#CFB87C", lwd=3)
```



In [10]:

```
#1.c
lr = loess(y ~ x, train_sim, span = 0.5) #loess fit to be used late
r

ggplot(train_sim, aes(x = x, y = y))+
    geom_point() +
    geom_smooth(color = "#CFB87C", span = 0.5)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In [11]:
```
#1.d
yhat_kr = ksmooth(train_sim$x, train_sim$y, kernel = "normal", 40,
x.points = test_sim$x)
cat("The MSPE for kernel regression is", mean((test_sim$y - yhat_k
r$y)^2),".")

yhat_ss = predict(sim.smooth, x = test_sim$x);
cat("The MSPE for smoothing spline regression is",mean((test_sim$y
- yhat_ss$y)^2), ".")
yhat_loess = predict(lr, newdata = test_sim$x);
cat("The MSPE for loess regression is", mean((test_sim$y - yhat_loe
ss)^2), ".")
```

```
The MSPE for kernel regression is 0.6674404 .The MSPE for smoothing
spline regression is 0.1443584 .The MSPE for loess regression is 0.1
422285 .
```

Again, it appears that the loess fit is best.