

C3M3_peer_review

April 29, 2025

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 contains 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 .

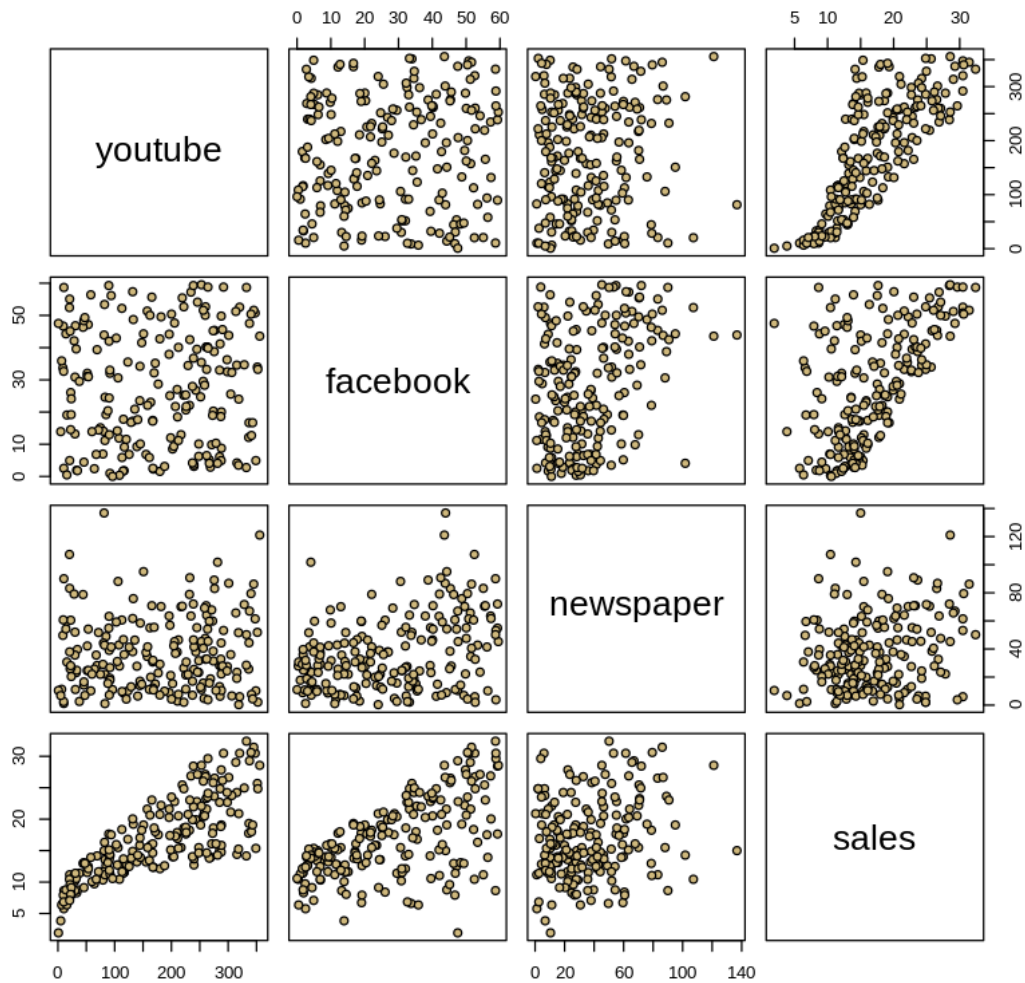
The advertising data treat “a company selling product P ” as the statistical unit, and “all companies selling product P ” as the population. We assume that the $n = 200$ companies in the dataset were chosen at random from the population (a strong assumption!).

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

```
[7]: # 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

Marketing Data



```
[8]: 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 indices 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.

```
[24]: set.seed(7240875)
bandwidth <- 10
colnames(marketing)

sales <- marketing$sales
youtube <- marketing$youtube

k.reg <- ksmooth(x = youtube, y = sales, "normal", bandwidth = bandwidth)

ggplot(train_marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  stat_smooth(method = "kernel", bandwidth = bandwidth, se = FALSE) +
  ggtitle("Kernel Regression on Sales vs. YouTube")

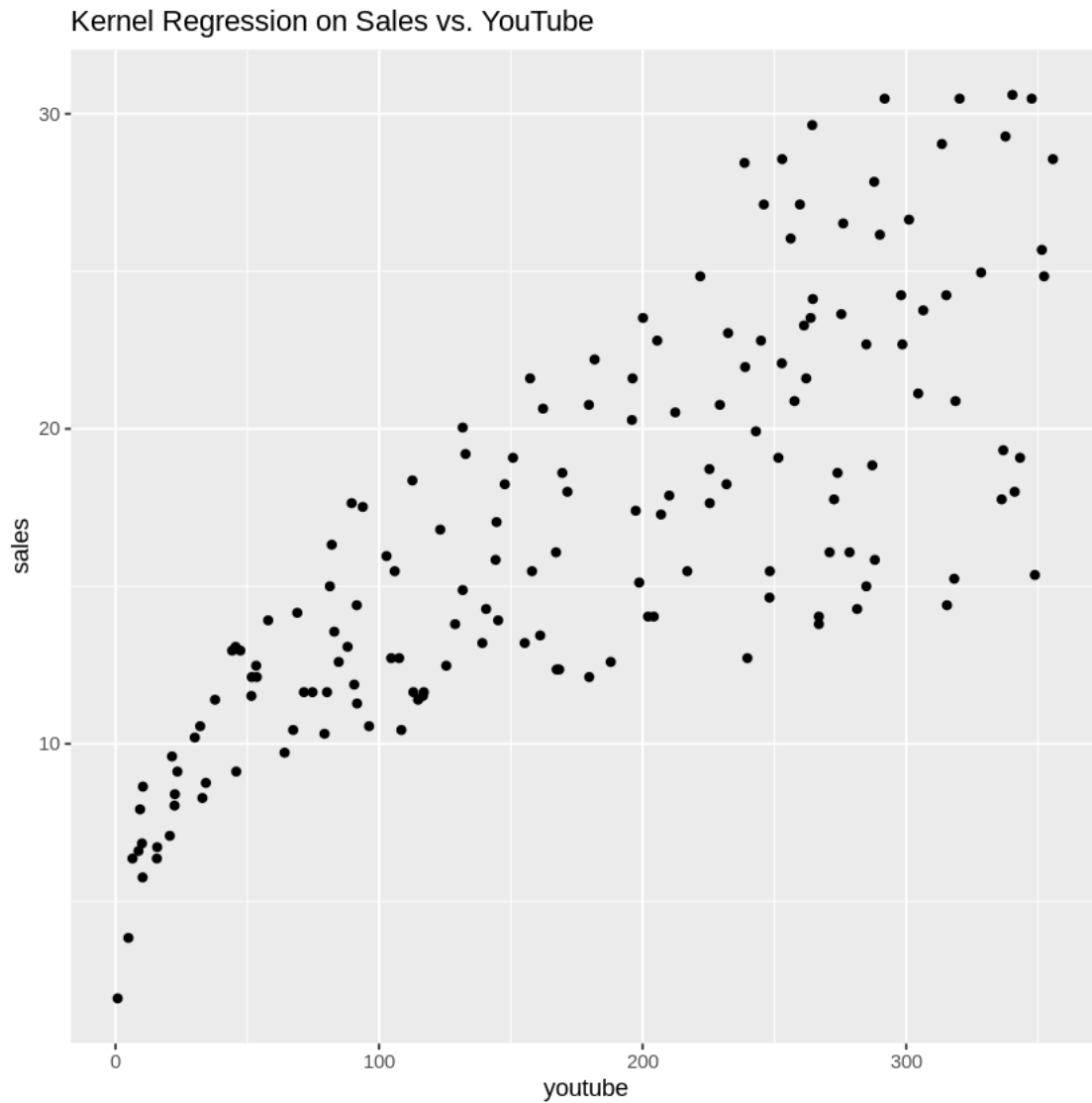
ksmooth_fit <- ksmooth(
  x = train_marketing$youtube,
  y = train_marketing$sales,
  kernel = "normal",
  bandwidth = bandwidth
)

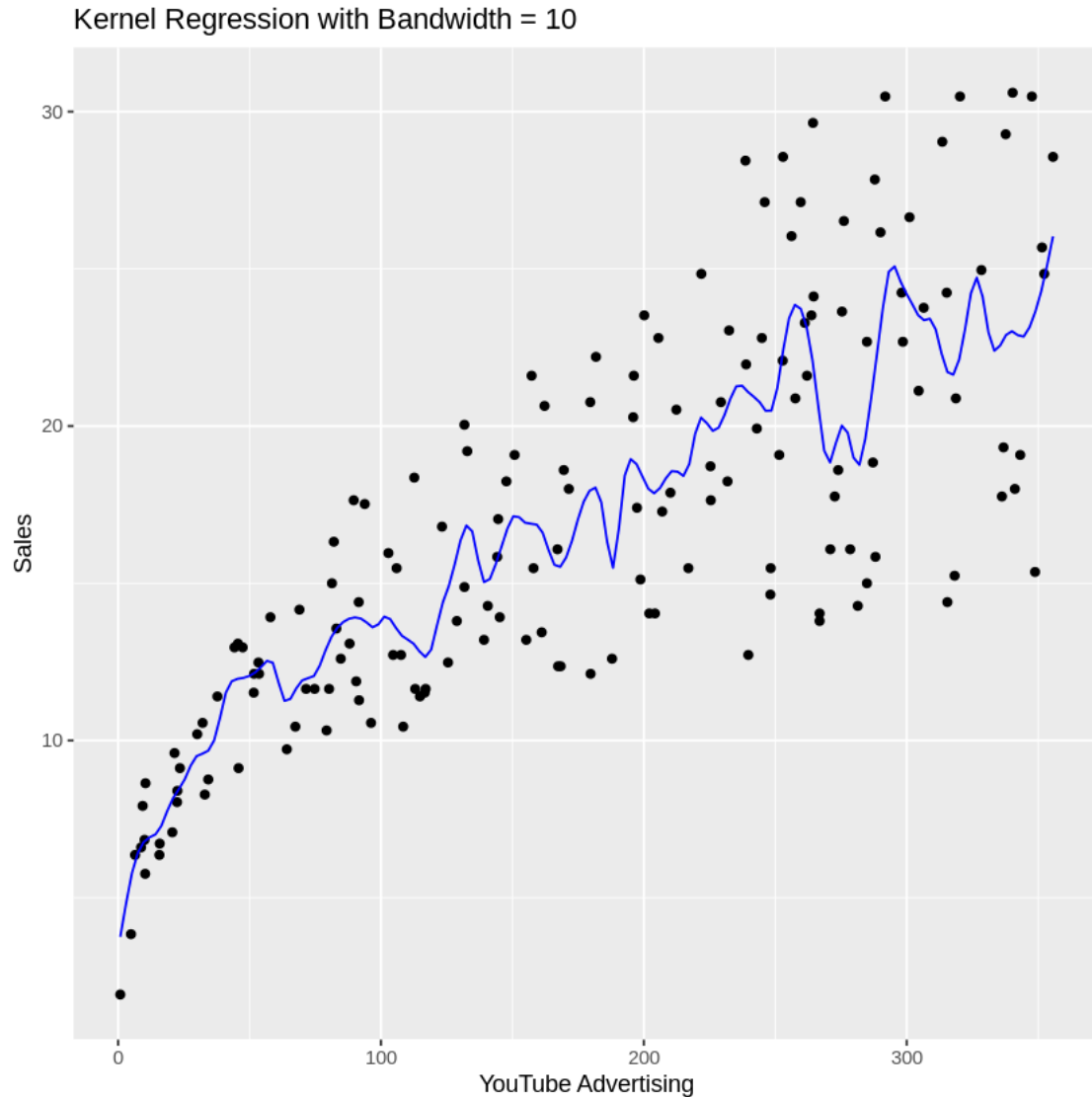
ggplot(train_marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  geom_line(data = data.frame(youtube = ksmooth_fit$x, sales = ksmooth_fit$y),
    aes(x = youtube, y = sales),
    color = "blue") +
  ggtitle(paste("Kernel Regression with Bandwidth =", bandwidth)) +
  xlab("YouTube Advertising") +
  ylab("Sales")
```

1. 'youtube' 2. 'facebook' 3. 'newspaper' 4. 'sales'

```
Warning message:  
"Ignoring unknown parameters: bandwidth"  
`geom_smooth()` using formula 'y ~ x'
```

```
Warning message:  
"Computation failed in `stat_smooth()`:  
unused arguments (data = data, weights = weight)"
```





A bandwidth of 10 overlays the data so that it captures the underlying pattern but does not show too much variance

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.

```
[30]: ggplot(train_marketing, aes(x = youtube, y = sales)) +  
  geom_point() +  
  ggtitle("Sales vs YouTube Advertising") +  
  xlab("YouTube Advertising") +  
  ylab("Sales")
```

```

smooth_param <- 0.7

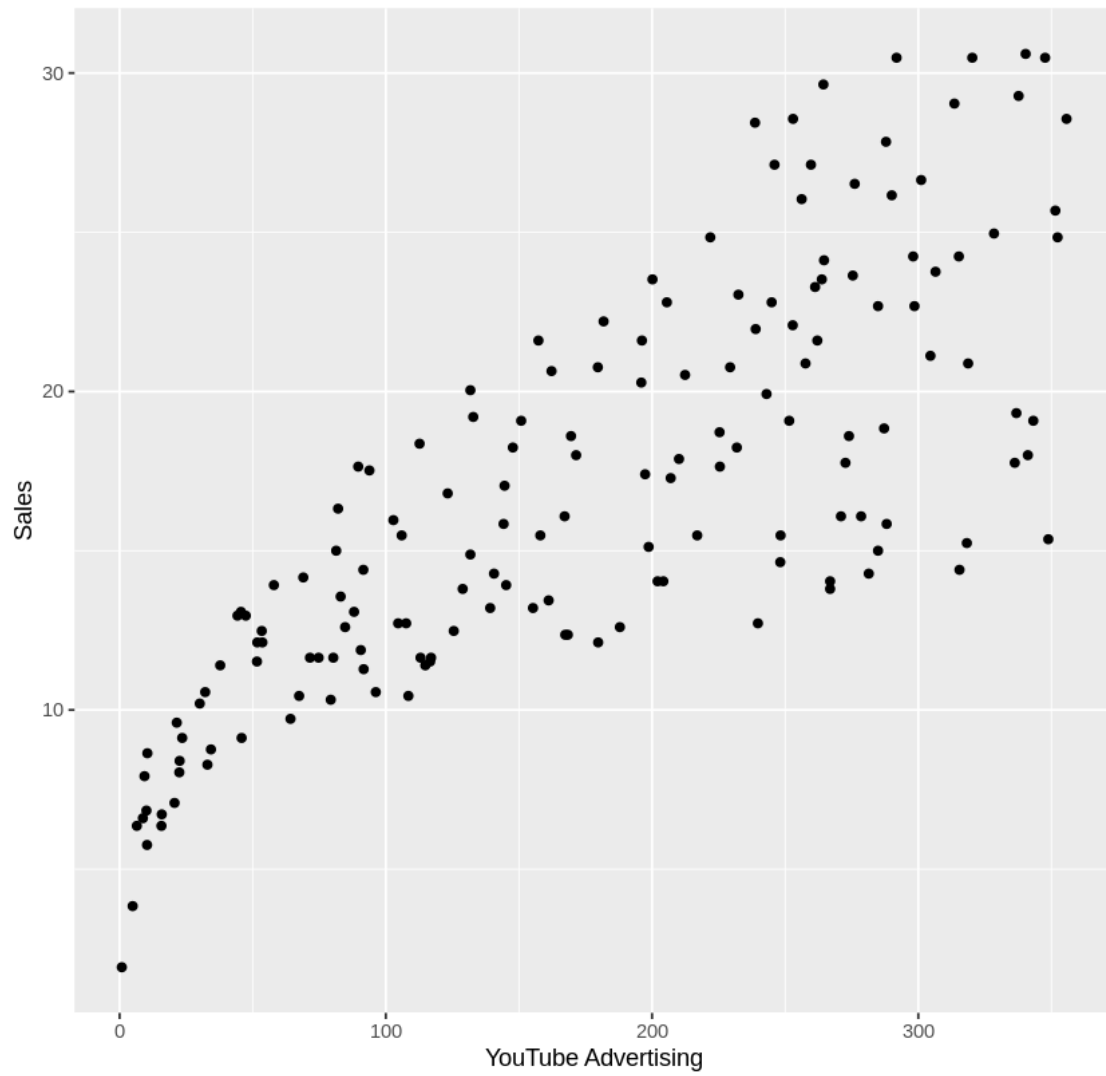
spline_fit <- smooth.spline(x = train_marketing$youtube, y =
  ↪train_marketing$sales, spar = smooth_param)

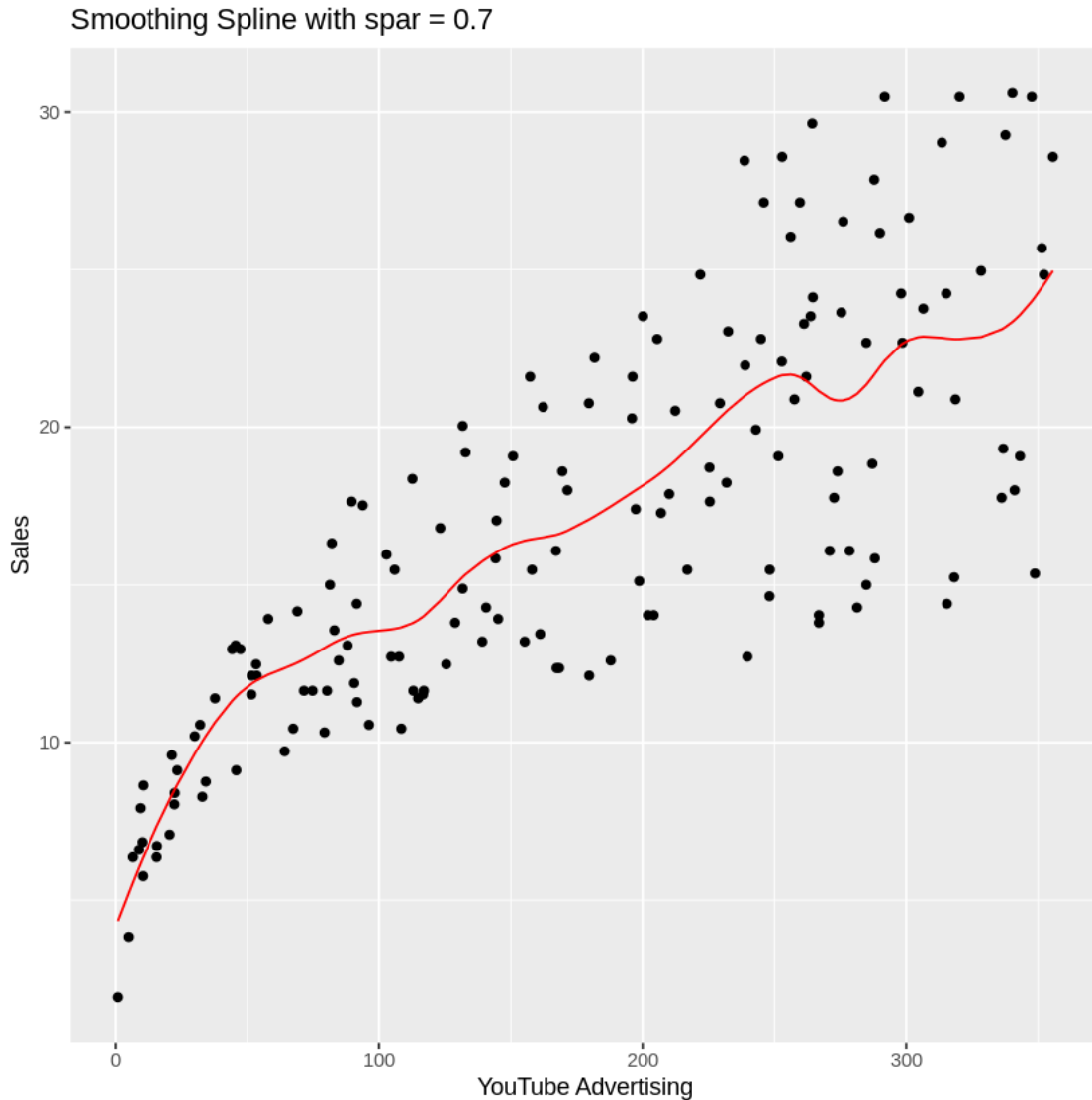
predicted_values <- predict(spline_fit, x = train_marketing$youtube)

ggplot(train_marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  geom_line(data = data.frame(youtube = predicted_values$x, sales =
  ↪predicted_values$y),
    aes(x = youtube, y = sales),
    color = "red") +
  ggtitle(paste("Smoothing Spline with spar =", smooth_param)) +
  xlab("YouTube Advertising") +
  ylab("Sales")

```

Sales vs YouTube Advertising





A smoothing spline of 0.7 provides a depiction of the relationship between sales and youtube without smoothing too much.

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.

```
[42]: p <- ggplot(train_marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  ggtitle("Sales vs YouTube Advertising") +
  xlab("YouTube Advertising") +
```

```

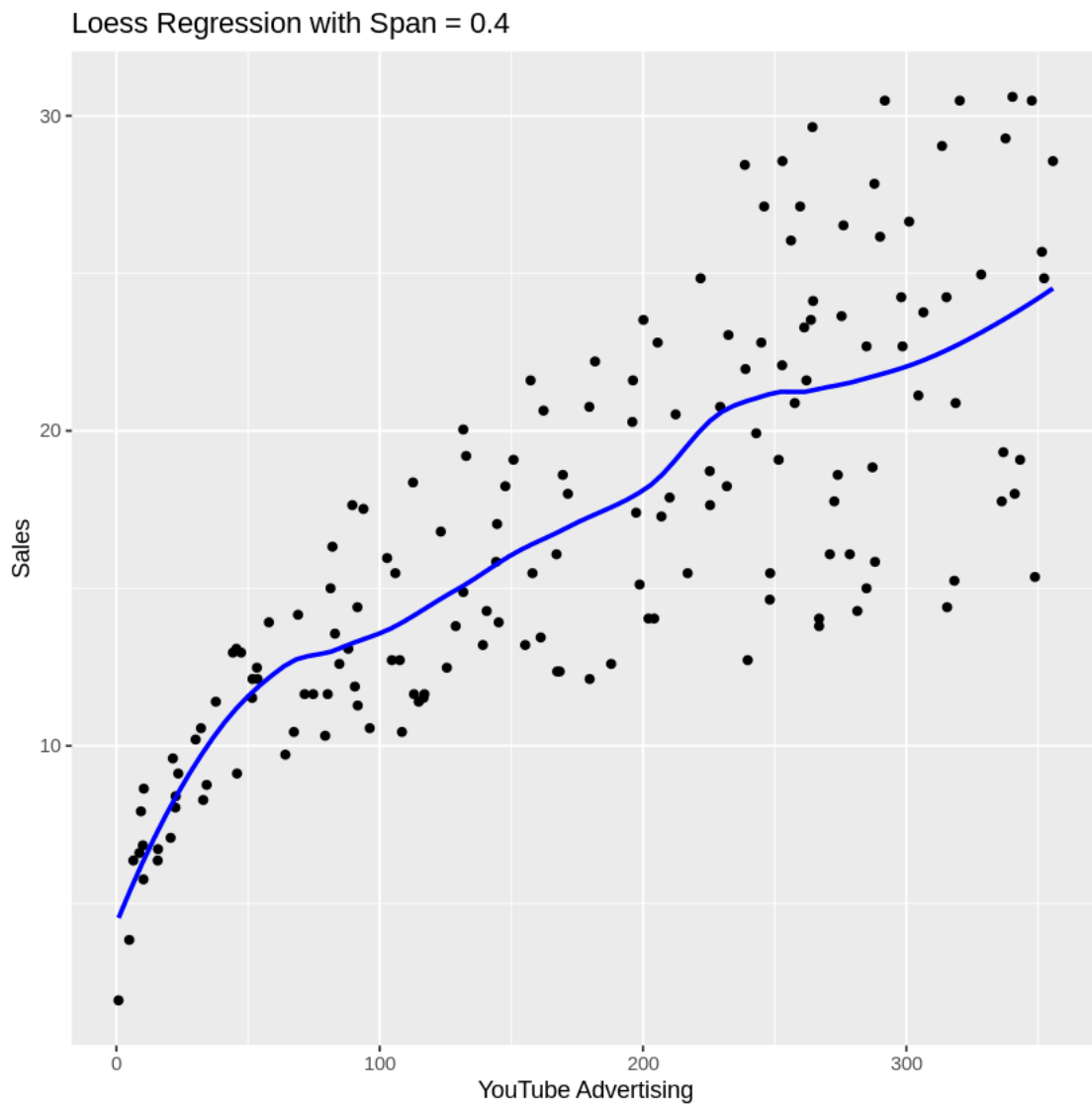
  ylab("Sales")

  span_value <- 0.4

  p + geom_smooth(method = "loess", span = span_value, se = FALSE, color = "blue",
    ↪ "blue") +
    ggtitle(paste("Loess Regression with Span =", span_value))

```

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



A span value of 0.4 gives a good pattern of the data relationship without smoothing too much.

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

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

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

*Note that `ksmooth()` orders your designated `x.points`. Make sure to account for this in your MSPE calculation.

```
[43]: bandwidth <- 10
ksmooth_fit <- ksmooth(train_marketing$youtube, train_marketing$sales, kernel = "normal", bandwidth = bandwidth)

ksmooth_pred <- approx(ksmooth_fit$x, ksmooth_fit$y, xout = test_marketing$youtube)$y

spline_fit <- smooth.spline(x = train_marketing$youtube, y = train_marketing$sales, spar = 0.7)
spline_pred <- predict(spline_fit, x = test_marketing$youtube)$y

span_value <- 0.5
loess_fit <- loess(sales ~ youtube, data = train_marketing, span = span_value)
loess_pred <- predict(loess_fit, newdata = test_marketing)

observed_sales <- test_marketing$sales

mspe_ksmooth <- mean((observed_sales - ksmooth_pred)^2, na.rm = TRUE)

mspe_spline <- mean((observed_sales - spline_pred)^2, na.rm = TRUE)

mspe_loess <- mean((observed_sales - loess_pred)^2, na.rm = TRUE)

cat("MSPE of Kernel Regression:", mspe_ksmooth, "\n")
cat("MSPE of Smoothing Spline:", mspe_spline, "\n")
cat("MSPE of Loess Regression:", mspe_loess, "\n")
```

```
best_model <- names(which.min(c(Kernel = mspe_ksmooth, Spline = mspe_spline,
↪Loess = mspe_loess)))
cat("The best model in terms of MSPE is:", best_model, "\n")
```

```
MSPE of Kernel Regression: 19.40443
MSPE of Smoothing Spline: 18.1893
MSPE of Loess Regression: 18.11483
The best model in terms of MSPE is: Loess
```

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.

```
[44]: set.seed(93710)

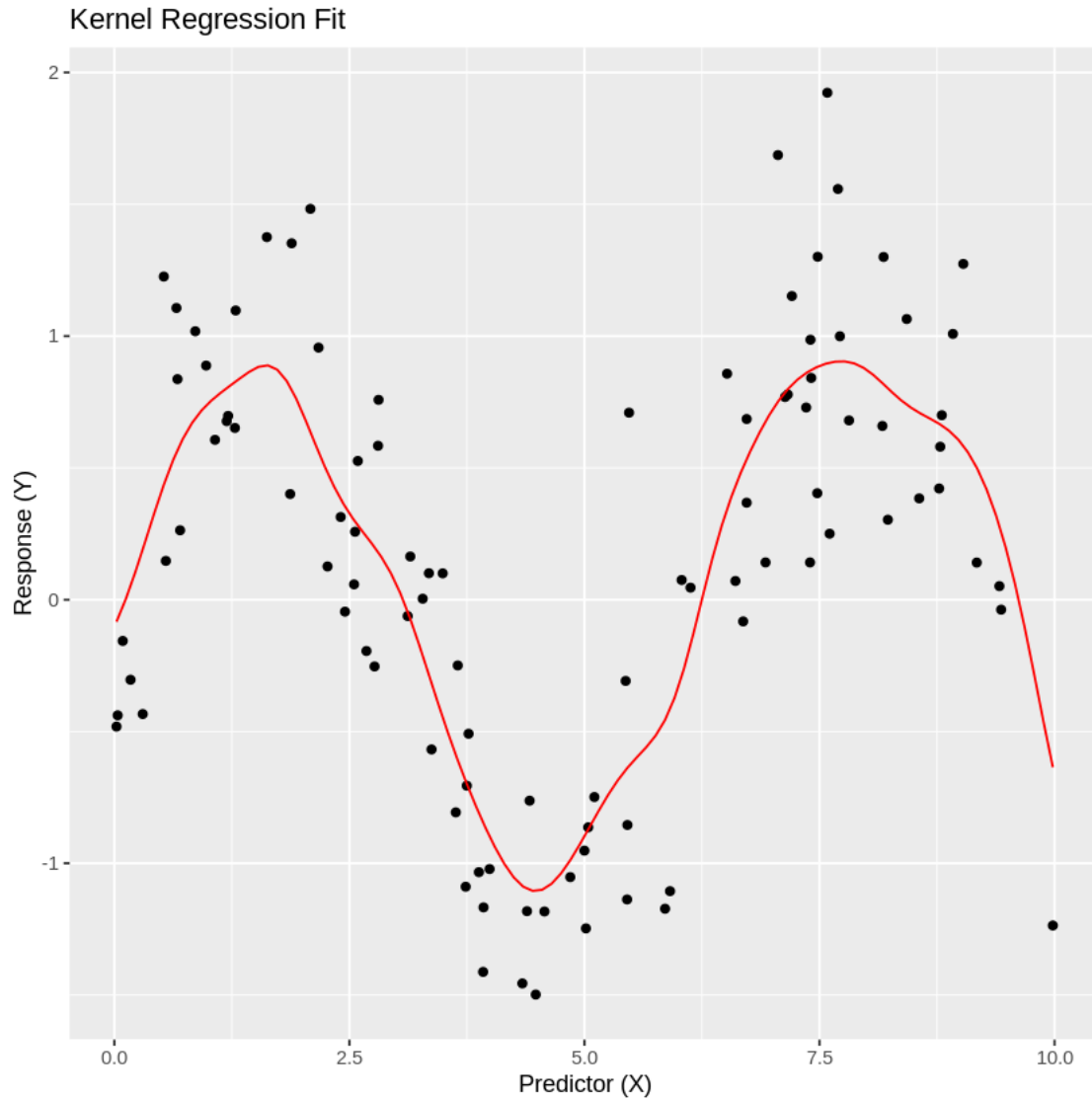
n <- 100
X <- runif(n, min = 0, max = 10)

noise <- rnorm(n, mean = 0, sd = 0.5)
Y <- sin(X) + noise

simulated_data <- data.frame(X = X, Y = Y)
```

```
[49]: #1.a
bandwidth <- 1
kernel_fit <- ksmooth(X, Y, kernel = "normal", bandwidth = bandwidth)

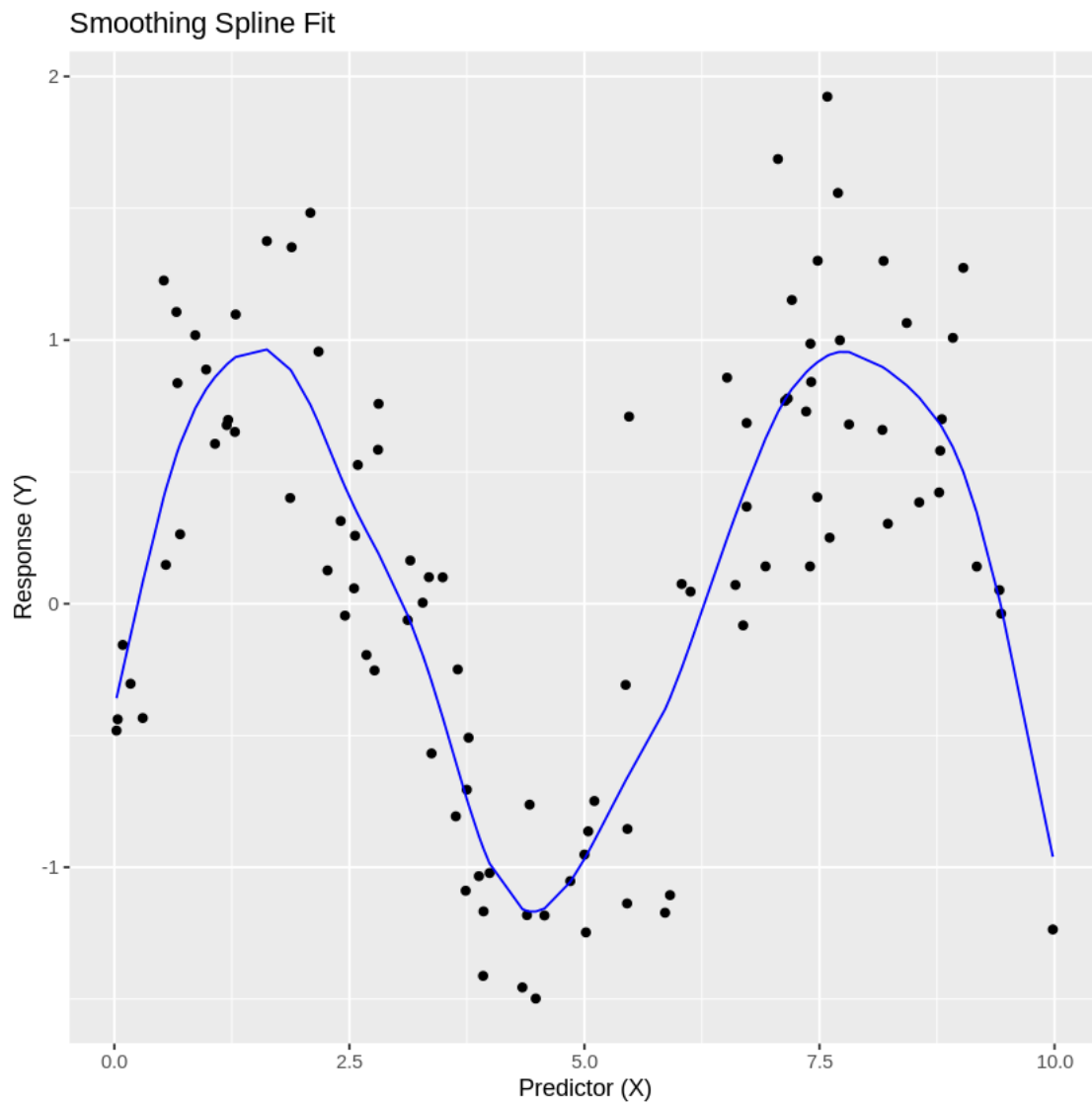
ggplot(simulated_data, aes(x = X, y = Y)) +
  geom_point() +
  geom_line(aes(x = kernel_fit$x, y = kernel_fit$y), color = "red") +
  ggtitle("Kernel Regression Fit") +
  xlab("Predictor (X)") +
  ylab("Response (Y)")
```



[50]: #1.b

```
spline_fit <- smooth.spline(x = X, y = Y, spar = 0.7)

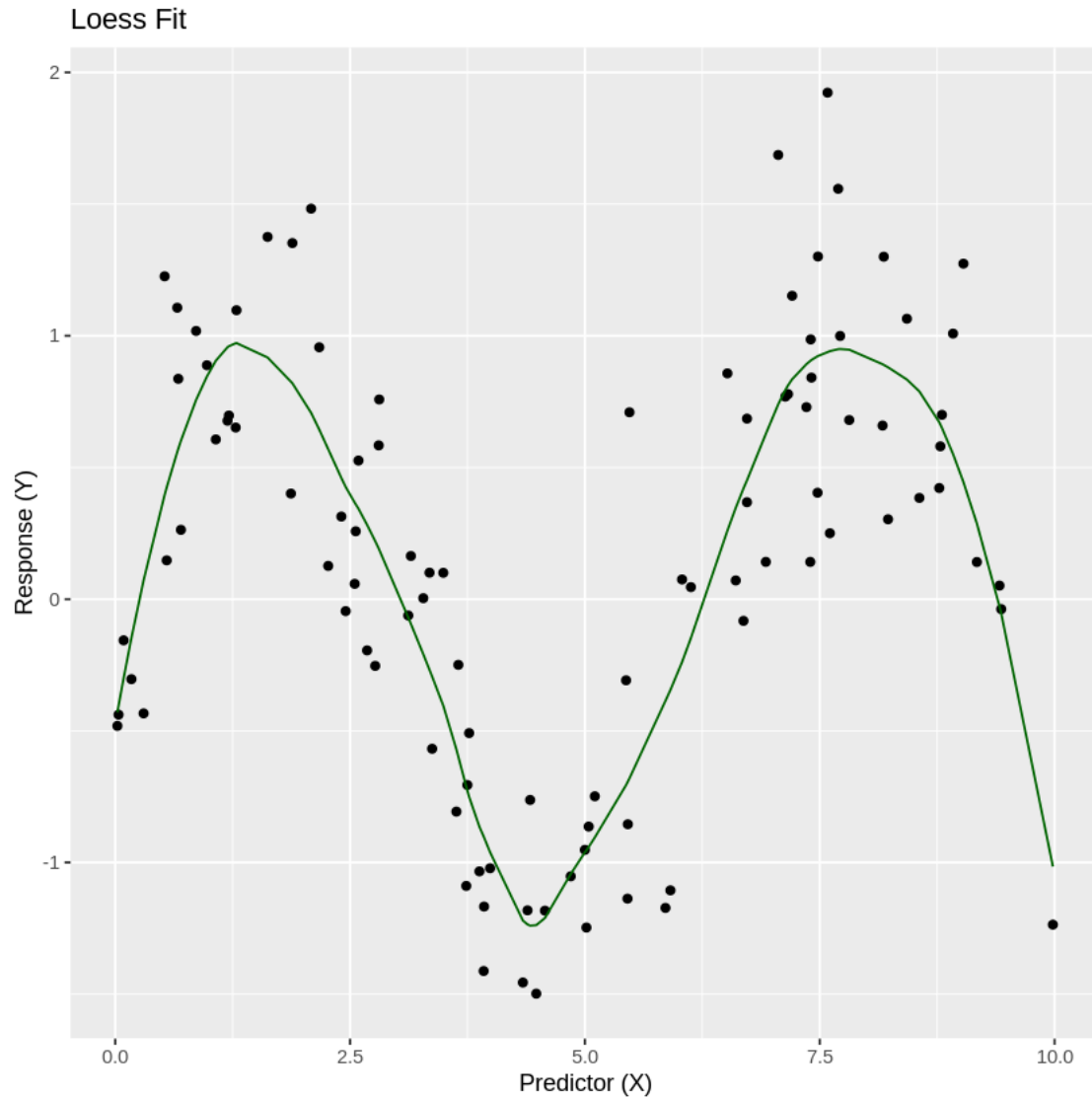
ggplot(simulated_data, aes(x = X, y = Y)) +
  geom_point() +
  geom_line(aes(x = spline_fit$x, y = spline_fit$y), color = "blue") +
  ggtitle("Smoothing Spline Fit") +
  xlab("Predictor (X)") +
  ylab("Response (Y)")
```



```
[58]: #1.c

span_value <- 0.35
loess_fit <- loess(Y ~ X, data = simulated_data, span = span_value)
loess_pred <- predict(loess_fit, newdata = simulated_data)

ggplot(simulated_data, aes(x = X, y = Y)) +
  geom_point() +
  geom_line(aes(x = X, y = loess_pred), color = "darkgreen") +
  ggtitle("Loess Fit") +
  xlab("Predictor (X)") +
  ylab("Response (Y)")
```



```
[59]: #1.d

set.seed(456)
train_indices <- sample(1:n, size = n * 0.8)
train_data <- simulated_data[train_indices, ]
test_data <- simulated_data[-train_indices, ]

kernel_pred <- approx(ksmooth(train_data$X, train_data$Y, kernel = "normal",
↪bandwidth = bandwidth),
                      xout = test_data$X)$y
```

```

spline_pred <- predict(smooth.spline(x = train_data$X, y = train_data$Y, spar = 0.7),
                        x = test_data$X)$y

loess_pred <- predict(loess(Y ~ X, data = train_data, span = span_value),
                      newdata = test_data)

mspe_kernel <- mean((test_data$Y - kernel_pred)^2, na.rm = TRUE)
mspe_spline <- mean((test_data$Y - spline_pred)^2, na.rm = TRUE)
mspe_loess  <- mean((test_data$Y - loess_pred)^2, na.rm = TRUE)

cat("Kernel Regression MSPE:", mspe_kernel, "\n")
cat("Smoothing Spline MSPE:", mspe_spline, "\n")
cat("Loess Regression MSPE:", mspe_loess, "\n")

best_model <- names(which.min(c(Kernel = mspe_kernel, Spline = mspe_spline,
                               Loess = mspe_loess)))
cat("The best model in terms of MSPE is:", best_model, "\n")

```

```

Kernel Regression MSPE: 0.3766276
Smoothing Spline MSPE: 0.3577602
Loess Regression MSPE: 0.3498132
The best model in terms of MSPE is: Loess

```