MSSC 5780 Homework 4

Due Date: November 4, 2022 11:59 PM Henrique Medeiros Dos Reis

1 Homework Instruction and Requirement

- Homework 5 covers course materials of Week 1 to 11.
- Please submit your work in **one PDF** file including all parts to **D2L > Assessments > Dropbox**. *Multiple files or a file that is not in pdf format are not allowed.*
- In your homework, please number and answer questions **in order**.
- Your entire work on Statistical Computing and Data Analysis should be completed by any word processing software (Microsoft Word, Google Docs, (R)Markdown, LaTex, etc) and your preferred programming language. Your document should be a PDF file.
- It is your responsibility to let me understand what you try to show. If you type your answers, make sure there are no typos. I grade your work based on *what you show, not what you want to show*. If you choose to handwrite your answers, write them neatly. If I can't read your sloppy handwriting, your answer is judged as wrong.

2 Statistical Computing and Data Analysis

Please perform a data analysis using `R` or your preferred language. **Any results should be generated by computer outputs, and your work should be done entirely by a computer. Handwriting is not allowed. Relevant code should be attached.**

2.1 Polynomial Regression

A solid-fuel rocket propellant loses weight after it is produced. The following data are available:

Months since Production, x	Weight Loss y (kg)
0.25	1.42
0.50	1.39
0.75	1.55
1.00	1.89
1.25	2.43
1.50	3.15
1.75	4.05
2.00	5.15
2.25	6.43
2.50	7.89

1. Fit a second-order polynomial $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$ to the data.

   Solution:

```
x<-seq(00.25,2.5, 0.25)
y<-c(1.42,1.39,1.55,1.89,2.43,3.15,4.05,5.15,6.43,7.89)
s_o_poly<-lm(y~x+I(x^2), data=data.frame(x,y))
```

2. Test for significance of regression.

   Solution:

```
summary(s_o_poly)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = data.frame(x, y))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.005364 -0.002727  0.001045  0.002409  0.003273
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.633000   0.004196   389.2  < 2e-16 ***
## x           -1.232182   0.007010  -175.8 5.09e-14 ***
## I(x^2)       1.494545   0.002484   601.6  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.003568 on 7 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.859e+06 on 2 and 7 DF,  p-value: < 2.2e-16
```

We can see that the f-statistic is huge, while the p-value is equal to 0, so this model is significant.

3. Test the hypothesis $H_0 : \beta_2 = 0$. Comment on the need for the quadratic term in this model.
   Solution:

```
summary(s_o_poly)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = data.frame(x, y))
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.005364 -0.002727  0.001045  0.002409  0.003273
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.633000   0.004196   389.2  < 2e-16 ***
## x           -1.232182   0.007010  -175.8 5.09e-14 ***
## I(x^2)       1.494545   0.002484   601.6  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.003568 on 7 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.859e+06 on 2 and 7 DF,  p-value: < 2.2e-16
```

Looking at summary, we can see that the p-value is essentially 0, and the F-statistic is huge, we can reject H_0, since β_2 is not 0, and the two regressions are not the same.

4. Are there any potential hazards in extrapolating with this model?

   Solution:

Since the order of this model is 2, the data outside of the range we have for X can be completely different from what we estimated.

5. Compute the R-student residuals for the second-order model. Analyze the residuals and comment on the adequacy of the model.

   Solution:

```
r_student<-rstudent(s_o_poly)
sort(r_student, decreasing = TRUE)
```

```
##          9          8          6          1          5          3          2
##  1.0955578  0.9847982  0.7878386  0.7160016  0.6071164  0.1307168 -0.1670682
##          7         10          4
## -1.2010436 -1.8006985 -2.0056738
```

They seem good, all are basically between 2 and -2, but point number 4, which is -2.00567, which is just a little smaller than -2

6. Fit a second-order model $y = \beta_0 + \beta_1 z + \beta_2 z^2 + \epsilon$ to the data, where $z = x - \bar{x}$, i.e., z is the centered x.

   Solution:

```
z <- x - mean(x)
(centered_lm <- lm(y ~ z+ I(z ^ 2), data = data.frame(z,y)))
```

```
##
## Call:
## lm(formula = y ~ z + I(z^2), data = data.frame(z, y))
##
## Coefficients:
## (Intercept)            z       I(z^2)
##       2.764        2.878        1.495
```

7. Construct the design matrix $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 \end{bmatrix}$ of the model in (1), where $\mathbf{w}_1$ is the predictor $\mathbf{x}$ vector *after* unit length scaling and $\mathbf{w}_2$ is the predictor $\mathbf{x}^2$ vector *after* unit length scaling. That is, for each $i = 1, 2, \ldots, n$, $w_{1i} = \frac{x_i - \bar{x}}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$, $w_{2i} = \frac{d_i - \bar{d}}{\sqrt{\sum_{i=1}^{n}(d_i - \bar{d})^2}}$ where $d_i = x_i^2$. Similarly, construct the design matrix for the model in (6). Remember to use the predictor z, the centered version of x.

   Solution:

```
w_1 <- (x-mean(x))/sqrt(sum((x-mean(x))^2))
d <- x^2
w_2 <- (d-mean(d))/sqrt(sum((d-mean(d))^2))
W_mod1 <-cbind(w_1,w_2)

w_1_z <- (z-mean(z))/sqrt(sum((z-mean(z))^2))
d_z <- z^2
w_2_z <- (d_z-mean(d_z))/sqrt(sum((d_z-mean(d_z))^2))
W_mod2 <- cbind(w_1_z,w_2_z)
```

8. Compute $\mathbf{W}'\mathbf{W}$ of the model in (1) and (6). In fact, $\mathbf{W}'\mathbf{W}$ is the correlation matrix of predictors. Which model has higher correlation between predictors?

   Solution:

```
t(W_mod1)%*%W_mod1
```

```
##           w_1       w_2
## w_1 1.0000000 0.9745586
## w_2 0.9745586 1.0000000
```

```
t(W_mod2)%*%W_mod2
```

```
##       w_1_z w_2_z
## w_1_z     1     0
## w_2_z     0     1
```

The model 1 has way higher correlation between predictors.

9. Compute the inverse of $\mathbf{W}'\mathbf{W}$ of the model in (1) and (6). In fact, the diagonal elements of the inverse matrix measures the degree of ill-conditioning. Does the centering of x help alleviate the ill-conditioning problem?

   Solution:

```
solve(t(W_mod1)%*%W_mod1)
```

```
##           w_1       w_2
## w_1  19.90625 -19.39981
## w_2 -19.39981  19.90625
```

```
solve(t(W_mod2)%*%W_mod2)
```

```
##       w_1_z w_2_z
## w_1_z     1     0
## w_2_z     0     1
```

The degrees in the off diagonal is much worse, which means that the model is ill-conditioned. It does help to alleviate by a lot, we get just identity for the second model, which is perfect.

2.2 Nonlinearity Diagnostics

Consider the Canadian occupational-prestige data set `Prestige` in the **carData** package. Treat the variable `prestige` as the response, and `education`, `income` and `women` are the three predictors.

```
library(carData)
head(Prestige)
```

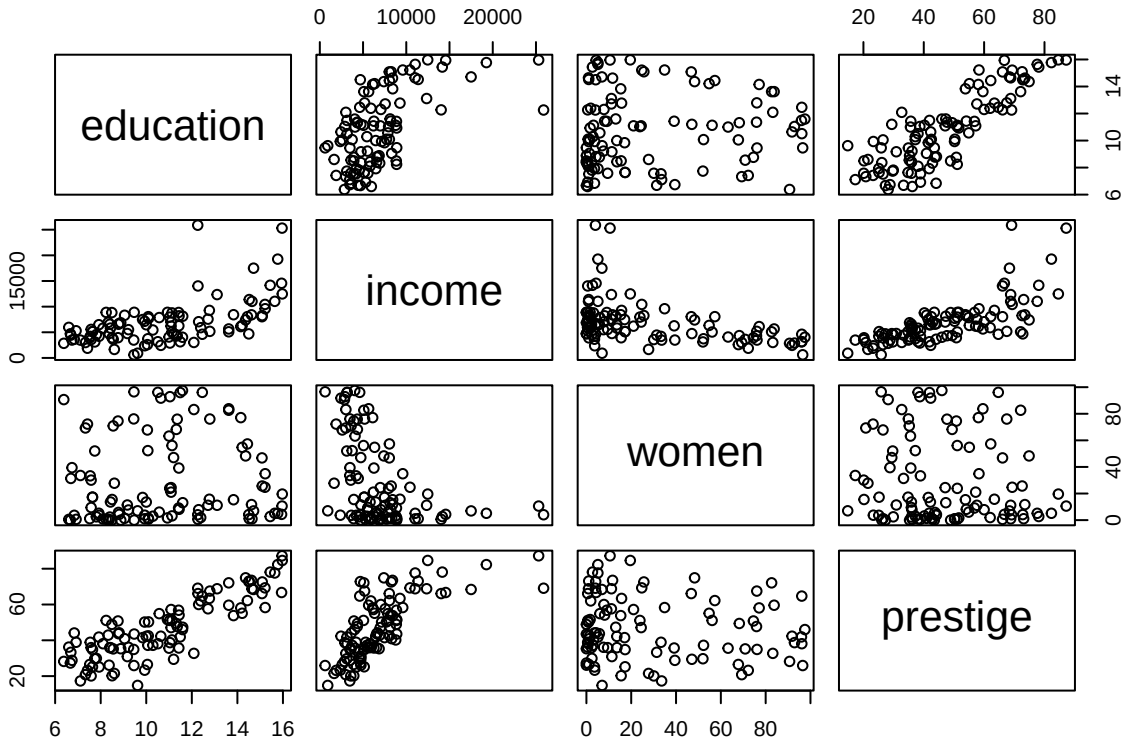
```
##                     education income women prestige census type
## gov.administrators      13.11  12351 11.16     68.8   1113 prof
## general.managers        12.26  25879  4.02     69.1   1130 prof
## accountants             12.77   9271 15.70     63.4   1171 prof
## purchasing.officers     11.42   8865  9.11     56.8   1175 prof
## chemists                14.62   8403 11.68     73.5   2111 prof
## physicists              15.64  11030  5.13     77.6   2113 prof
```

```
# We can see that the indexes are:
# prestige = 4
# education = 1
# income = 2
# women = 3
```

1. Generate the scatterplot matrix. Does the plot suggest any nonlinear relationship between the response and predictors or among predictors? Is the plot useful for detecting nonlinearity assumption of regression?

Solution:

```
pairs(Prestige[,1:4])
```

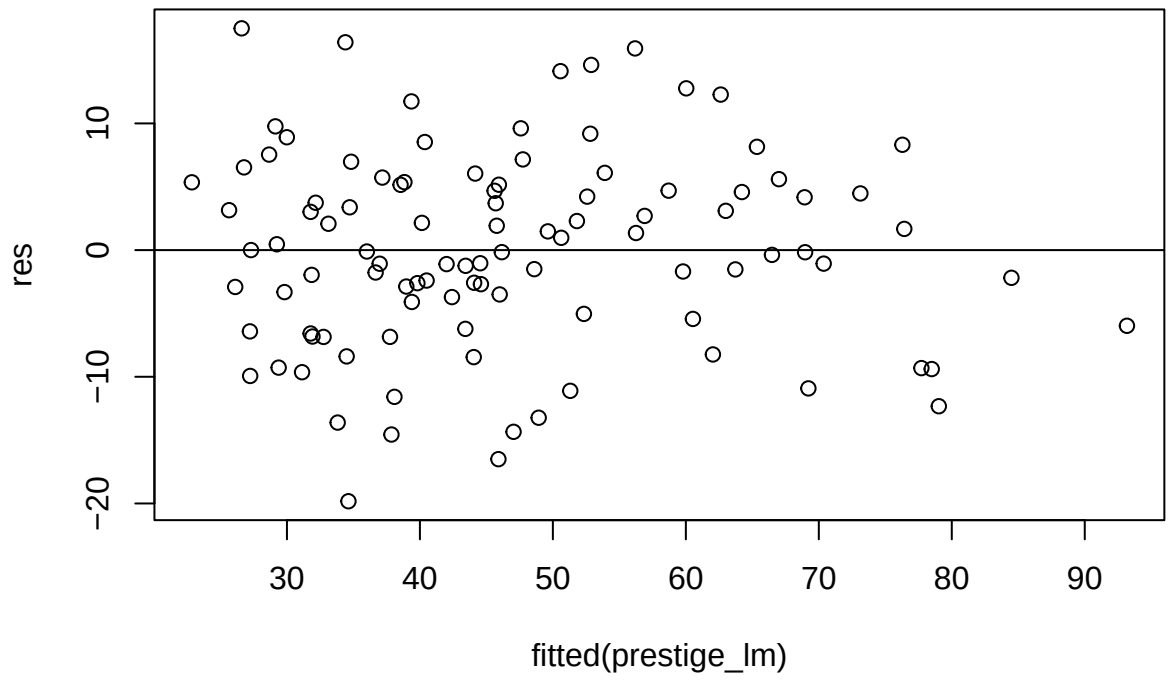


Simply scatterploting y against each x can lead to wrong conclusions, so we would need further testing.Since it only shows the marginal relationship between y and each x, without controlling the level of other regressors. However, it does suggests some nonlinear relation between women-prestige, and income-prestige.

2. Fit the linear regression model, and plot the residual plots. Comment the plot.

Solution:

```
prestige_lm <- lm(prestige ~ education + women + income, data = Prestige)
res <- resid(prestige_lm)
plot(fitted(prestige_lm), res)
abline(0,0)
```

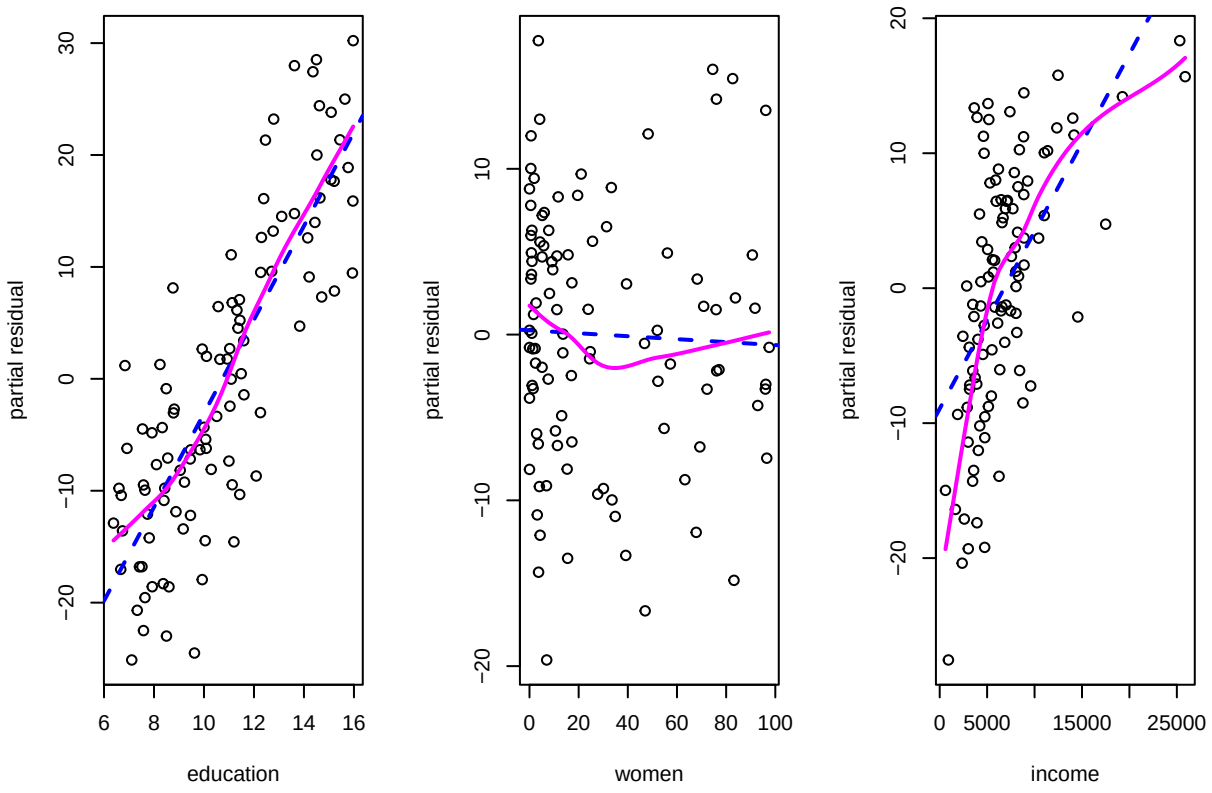


Our residuals seem to be a little to big for the model. Even though they may seem normally distributed, there may be some way to make this model a little better.

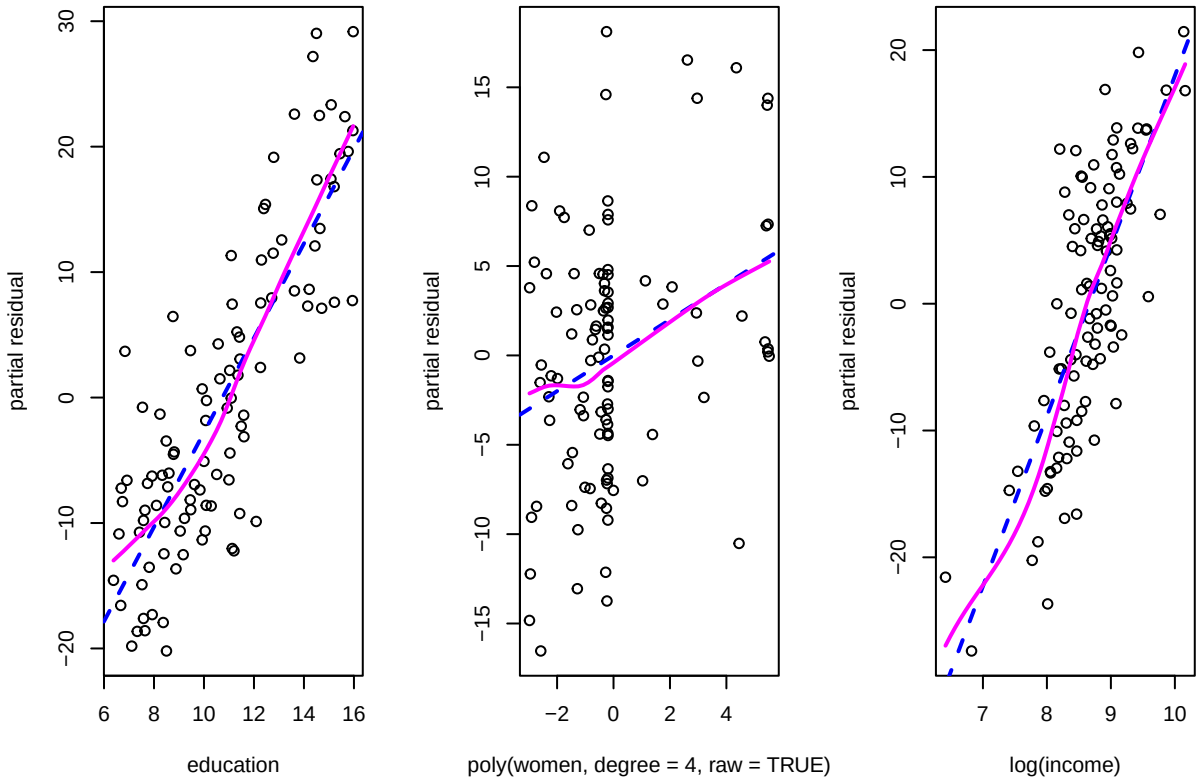
3. Generate partial residual plots (component-plus-residual plot). Based on the plot, transform any predictors if necessary, and refit the regression. Check the partial residual plot of the refitted model and confirm that the nonlinearity issue is fixed.

   Solution:

```
car::crPlots(prestige_lm, ylab = "partial residual", layout = c(1, 3), grid = FALSE, main = ""
```




```
trans_prestige_lm <- lm(prestige ~ education + poly(women,degree = 4,raw=TRUE) + log(income), d
car::crPlots(trans_prestige_lm, ylab = "partial residual", layout = c(1, 3), grid = FALSE, mai
```



After the transformations, the issue seems to be fixed, or at least improved by a lot.

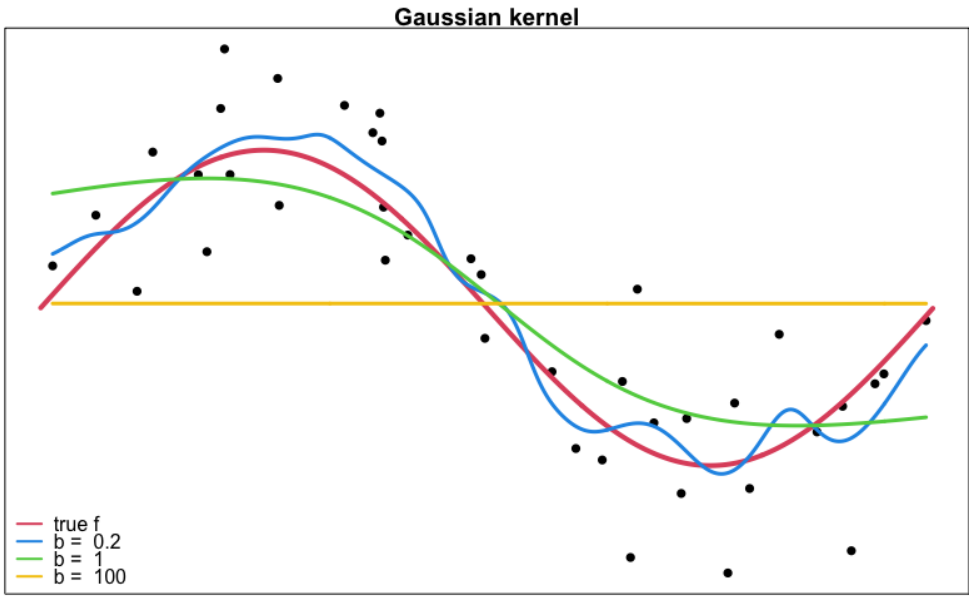
2.3 Kernel Methods

1. Generate the 40 data points $\{x_i, y_i\}_{i=1}^{40}$ from the model $y = f(x) + \epsilon$, where $f(x) = 2\sin(x)$ and $\epsilon \sim N(0, 1)$. Use `seq()` to generate the x_is.

   Solution:

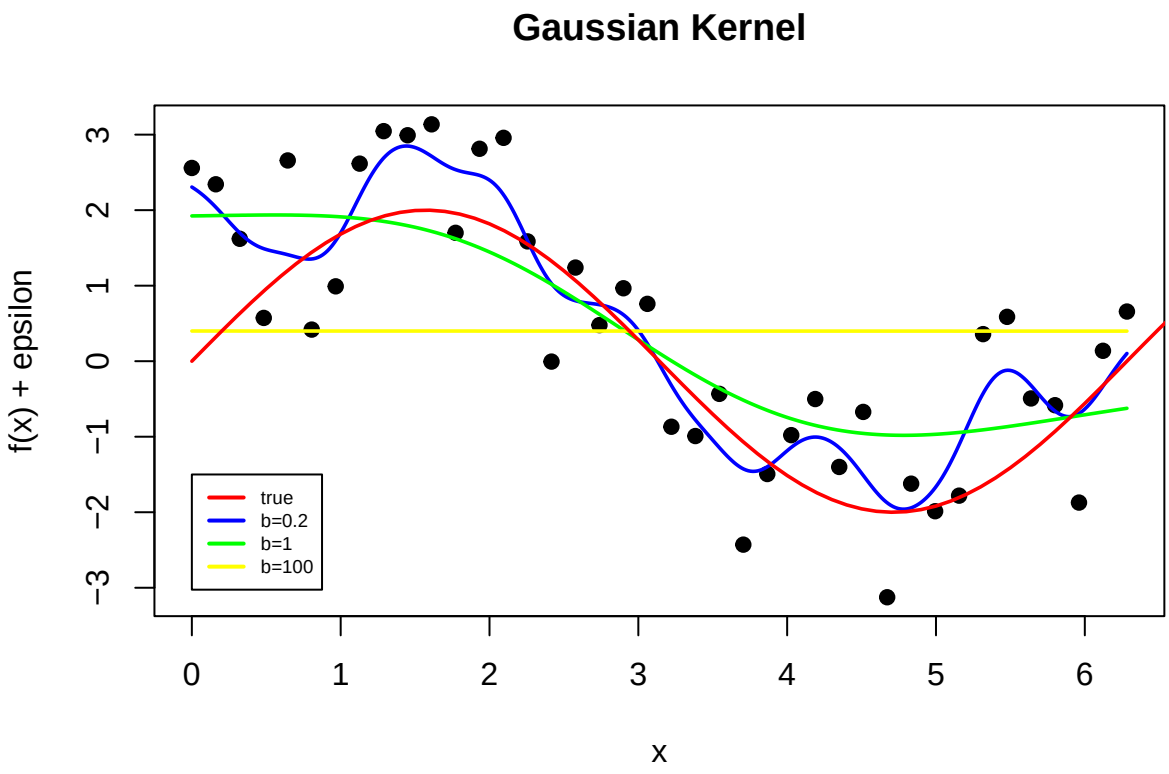
```
x_2_3<-seq(0,2*pi, length=40)
epsilon <- rnorm(40)
y_2_3<-2*sin(x_2_3)+epsilon
```

2. Use `KernSmooth::locpoly()` to generate Gaussian kernel smoothers with bandwidth 0.2, 1, and 100. Plot them as the figure shown below.



Solution:

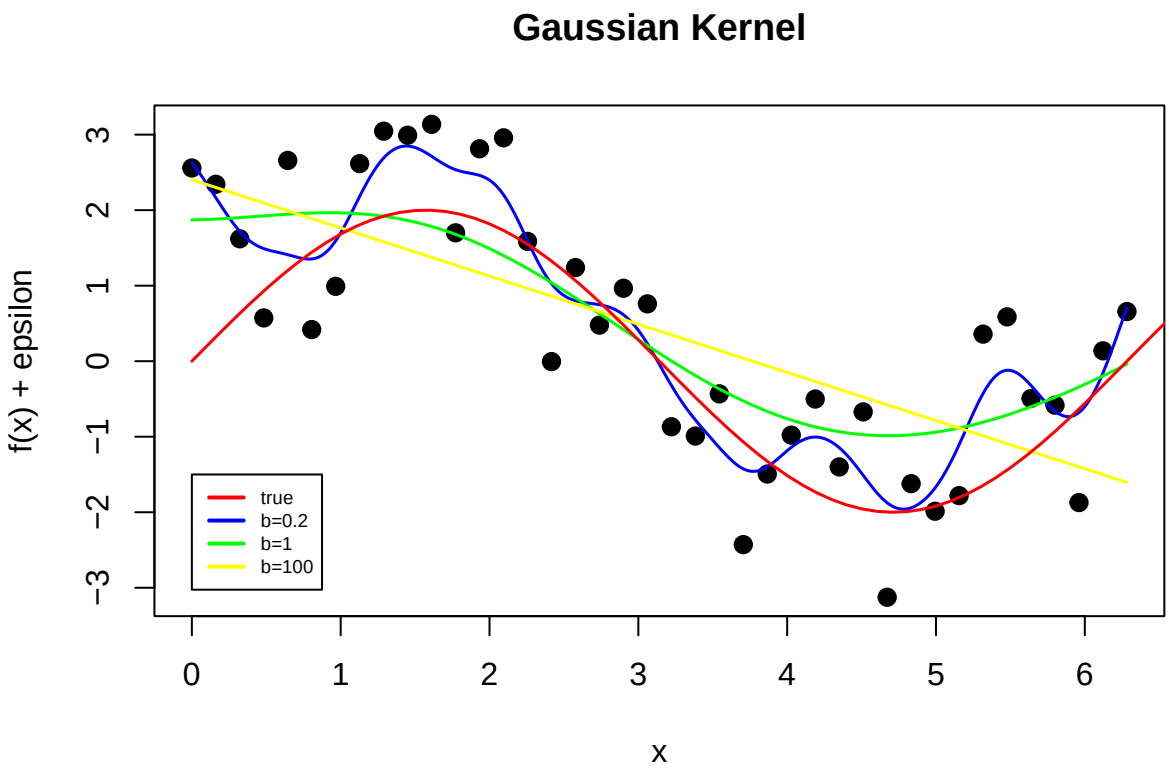
```
smooth_b_02<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 0, kernel = "normal", bandwidth = 0.2)
smooth_b_1<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 0, kernel = "normal", bandwidth = 1)
smooth_b_100<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 0, kernel = "normal", bandwidth = 100)

t<-seq(0,40,0.1)
t_y <- 2*sin(t)

plot(x_2_3,y_2_3, pch=19, xlab="x", ylab="f(x) + epsilon")
lines(smooth_b_02,col="blue", lwd=1.8)
lines(smooth_b_1,col="green", lwd=1.8)
lines(smooth_b_100,col="yellow", lwd=1.8)
lines(t,t_y, col="red", lwd=1.8)
title("Gaussian Kernel")
legend(0,-1.5,c("true","b=0.2","b=1","b=100"), lwd=c(2,2), col=c("red","blue","green", "yellow
```



3. Use `locpoly()` to fit the Gaussian local linear regressions (degree = 1) with bandwidth 0.2, 1, and 100. Generate the plot as the one in (2).

Solution:

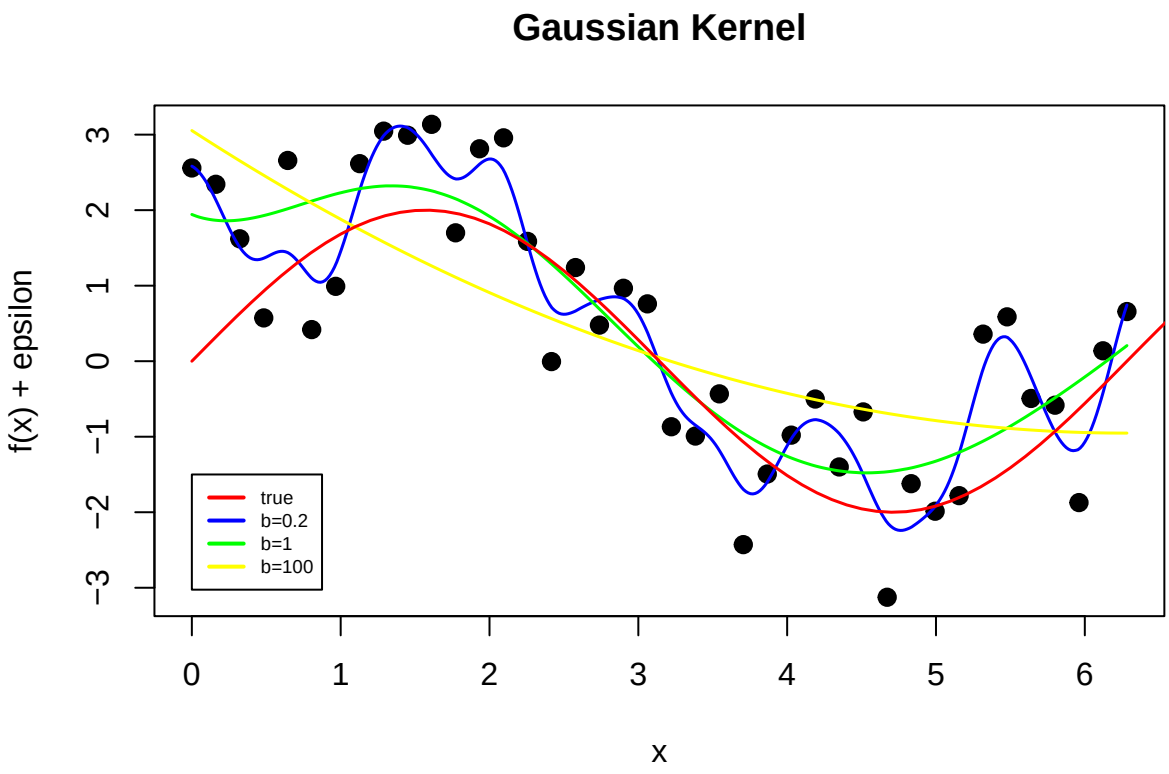
```
smooth_b_02<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 1, kernel = "normal", bandwidth = 0.2)
smooth_b_1<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 1, kernel = "normal", bandwidth = 1)
smooth_b_100<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 1, kernel = "normal", bandwidth = 100)

t<-seq(0,40,0.1)
t_y <- 2*sin(t)

plot(x_2_3,y_2_3, pch=19, xlab="x", ylab="f(x) + epsilon",cex=1.2)
lines(smooth_b_02,col="blue", lwd=1.5)
lines(smooth_b_1,col="green", lwd=1.5)
lines(smooth_b_100,col="yellow", lwd=1.5)
lines(t,t_y, col="red", lwd=1.5)
title("Gaussian Kernel")
legend(0,-1.5,c("true","b=0.2","b=1","b=100"), lwd=c(2,2), col=c("red","blue","green", "yellow
```



4. Use `locpoly()` to fit the Gaussian local quadratic regressions (degree = 2) with bandwidth 0.2, 1, and 100. Generate the plot as the one in (2).

Solution:

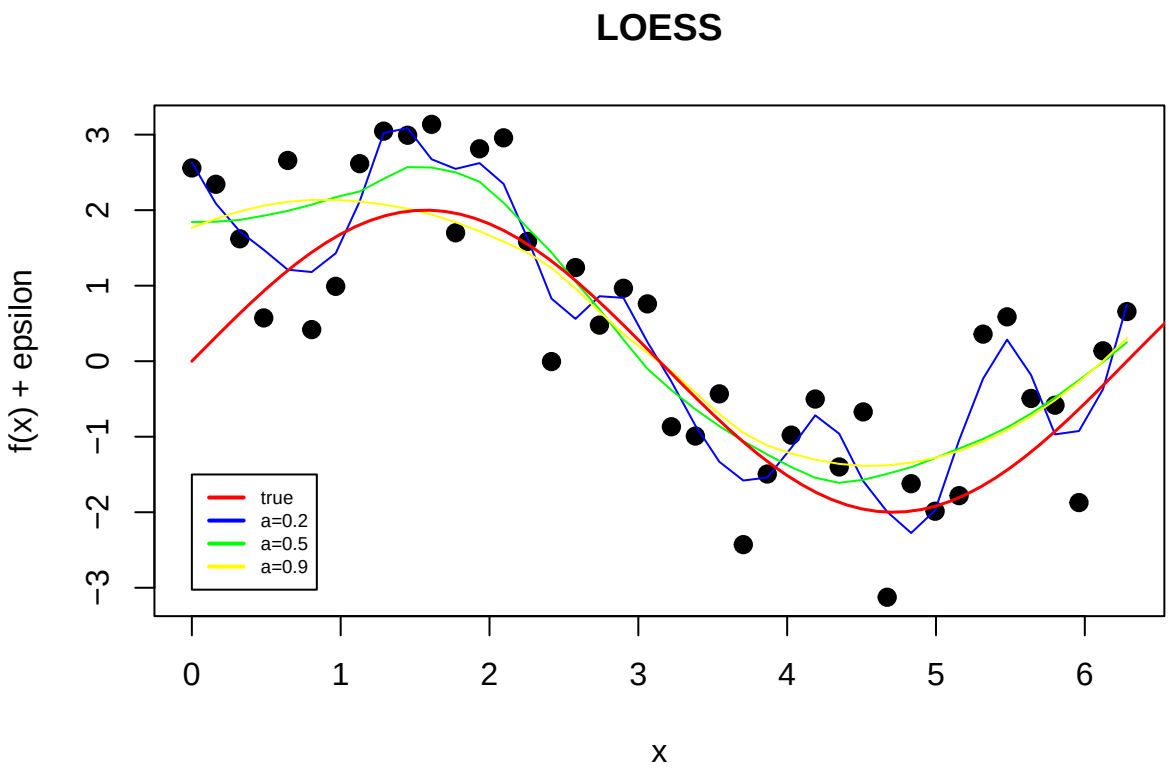
```
smooth_b_02<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 2, kernel = "normal", bandwidth = 0.2)
smooth_b_1<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 2, kernel = "normal", bandwidth = 1)
smooth_b_100<-KernSmooth::locpoly(x_2_3, y_2_3, degree = 2, kernel = "normal", bandwidth = 100)

t<-seq(0,40,0.1)
t_y <- 2*sin(t)

plot(x_2_3,y_2_3, pch=19, xlab="x", ylab="f(x) + epsilon",cex=1.2)
lines(smooth_b_02,col="blue", lwd=1.5)
lines(smooth_b_1,col="green", lwd=1.5)
lines(smooth_b_100,col="yellow", lwd=1.5)
lines(t,t_y, col="red", lwd=1.5)
title("Gaussian Kernel")
legend(0,-1.5,c("true","b=0.2","b=1","b=100"), lwd=c(2,2), col=c("red","blue","green", "yellow
```



5. Use `loess()` to fit the tricube local quadratic regressions with the span parameter $\alpha = 0.2, 0.5, 0.9$. Generate the plot as the one in (2).

Solution:

```
plot(x_2_3,y_2_3, pch=19, xlab="x", ylab="f(x) + epsilon",cex=1.2)
lines(x_2_3,loess(y_2_3 ~ x_2_3, span = 0.2)$fitted,col="blue")
lines(x_2_3,loess(y_2_3 ~ x_2_3, span = 0.5)$fitted,col="green")
lines(x_2_3,loess(y_2_3 ~ x_2_3, span = 0.9)$fitted,col="yellow")
lines(t,t_y, col="red", lwd=1.5)
title("LOESS")
legend(0,-1.5,c("true","a=0.2","a=0.5","a=0.9"), lwd=c(2,2), col=c("red","blue","green", "yell
```



6. Comment on your plots in (2)-(5), and discuss your findings. Which method you prefer?

Solution: It is possible to see that using Gaussian kernels with a large value of b is not the most efficient way to fit our data. While using a small value of b is not the best way to go either. In all the 3 plots using Gaussian Kernels, $b = 1$ gives the best result. Now talking about the degrees that should be used to fit this data, the degree being equal to 2 seems to give the best result, since the difference in amplitude from the true relationship to the Gaussian Kernel using $b = 1$ and degree$= 2$ seems to be the smallest among all the other ones. LOESS also gives a pretty good result, but it does not seem as good as the one in plot (4)