Statistical models : Homework 3

Priyanshi Shah and Sourabh Prakash

2023-02-03

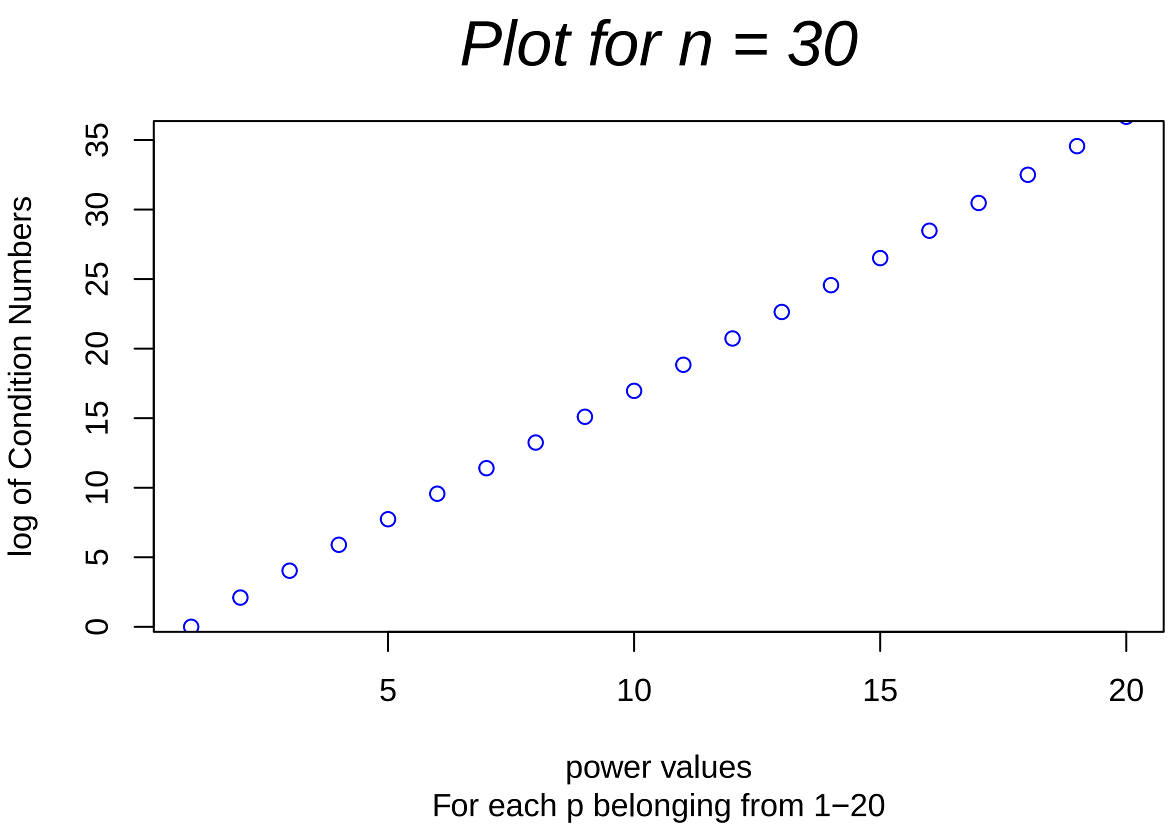
Question 1

Problem 1. (Is polynomial regression well-conditioned?) Consider the canonical design matrix for fitting a polynomial of degree p based on x1,. . . ,xn, meaning, X = (xji) for i = 1,. . . ,n and j = 0, . . . , p. Suppose the xi's are evenly distributed in (0, 1), for example, xi = i/(n + 1) for i = 1,. . . ,n. For each p = {1,. . . ,20} and each n = {30,50,100,200,500,1000}, compute the condition number of the design matrix. Produce a useful plot for visualizing the result of these computations. Offer brief comments.

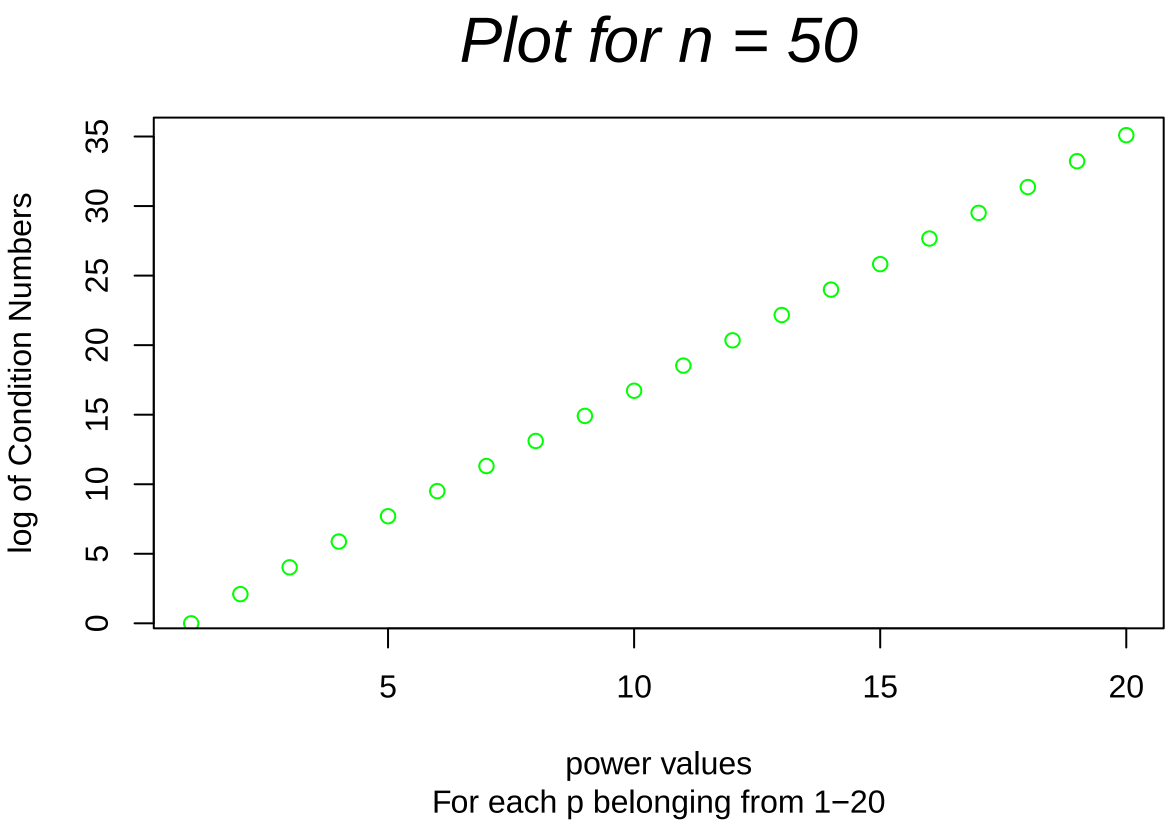
```r
fun <- function(row,col) row^(col-1)
final_list = c()
# Creating a function for condition number
condition_number = function(n, color="blue"){
  list_k_X = c()
  i = n
  #looping from 1 to 20
  for (j in 1:20){
    x = seq(from = 0, to = 1, by = (1/(i+1)))[2:i+1]
    rows = x
    cols = 1:j+1
    X = outer(rows,cols,FUN=fun)
    svd_X = svd(X)
    k_X = max(svd_X$d)/min(svd_X$d)
    list_k_X = append(list_k_X, log(k_X))
    #plot(list_k_X, col=color)
  }
  #Visualizing the plot
 plot(list_k_X, col=color, xlim=c(1,20), ylim = c(1,35),
      xlab = "power values", ylab = "log of Condition Numbers")
  title(main = paste('Plot for n =', n), sub = "For each p belonging from 1-20",
      cex.main = 2,   font.main= 3, col.main= "black")
  final_list = append(final_list,list_k_X)
  }

#plot.new()
```

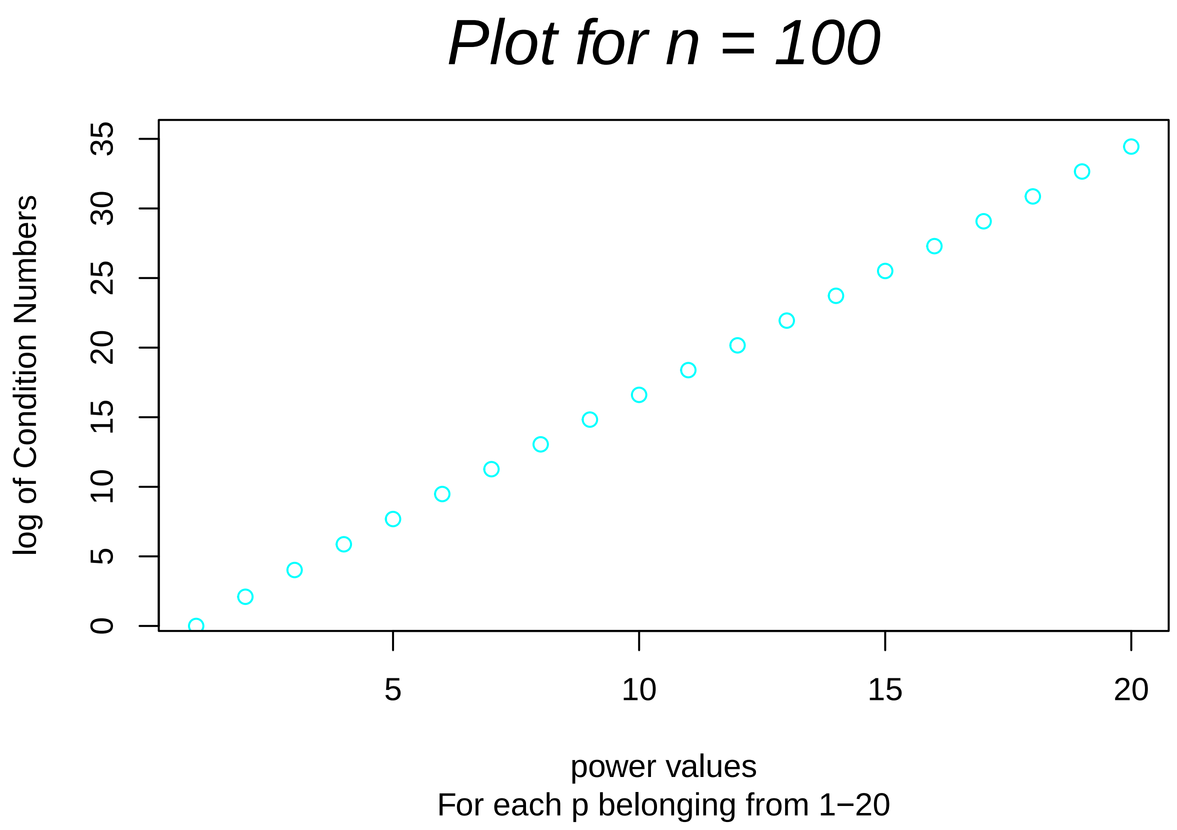
```r
condition_number(30, "blue")
```



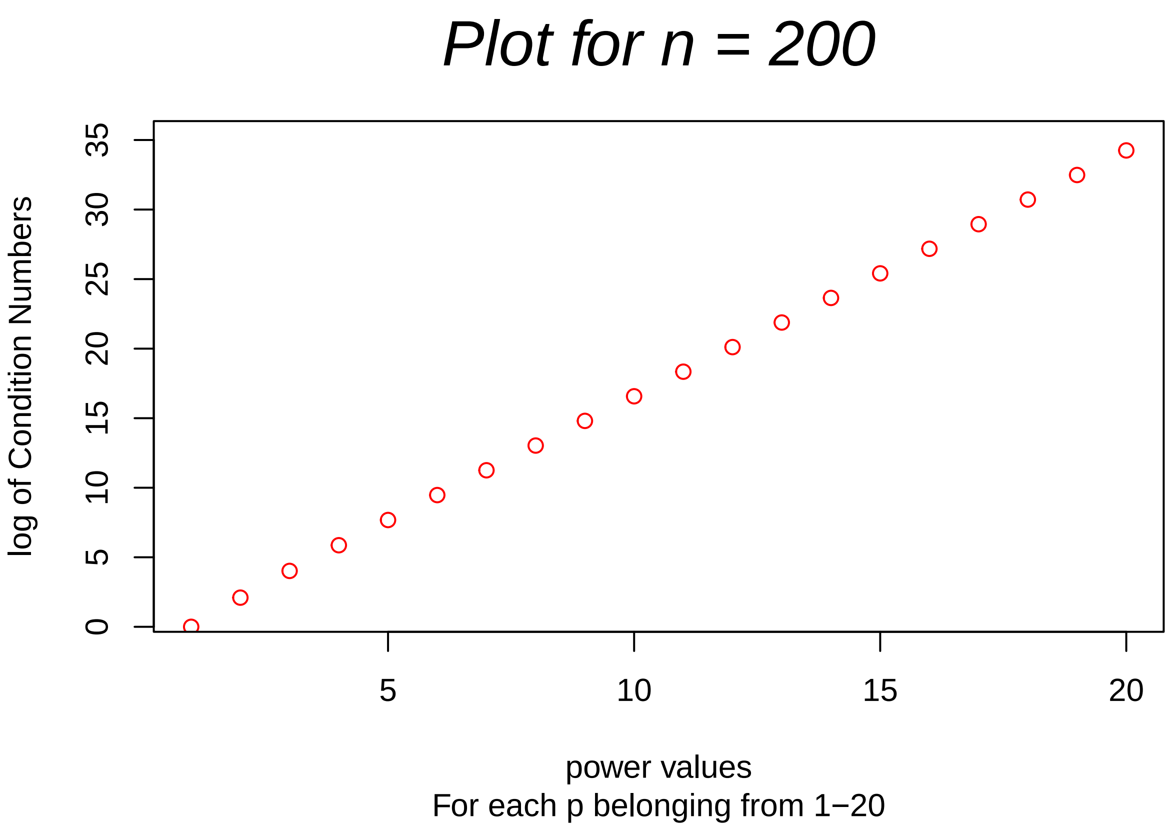
```
condition_number(50, "green")
```



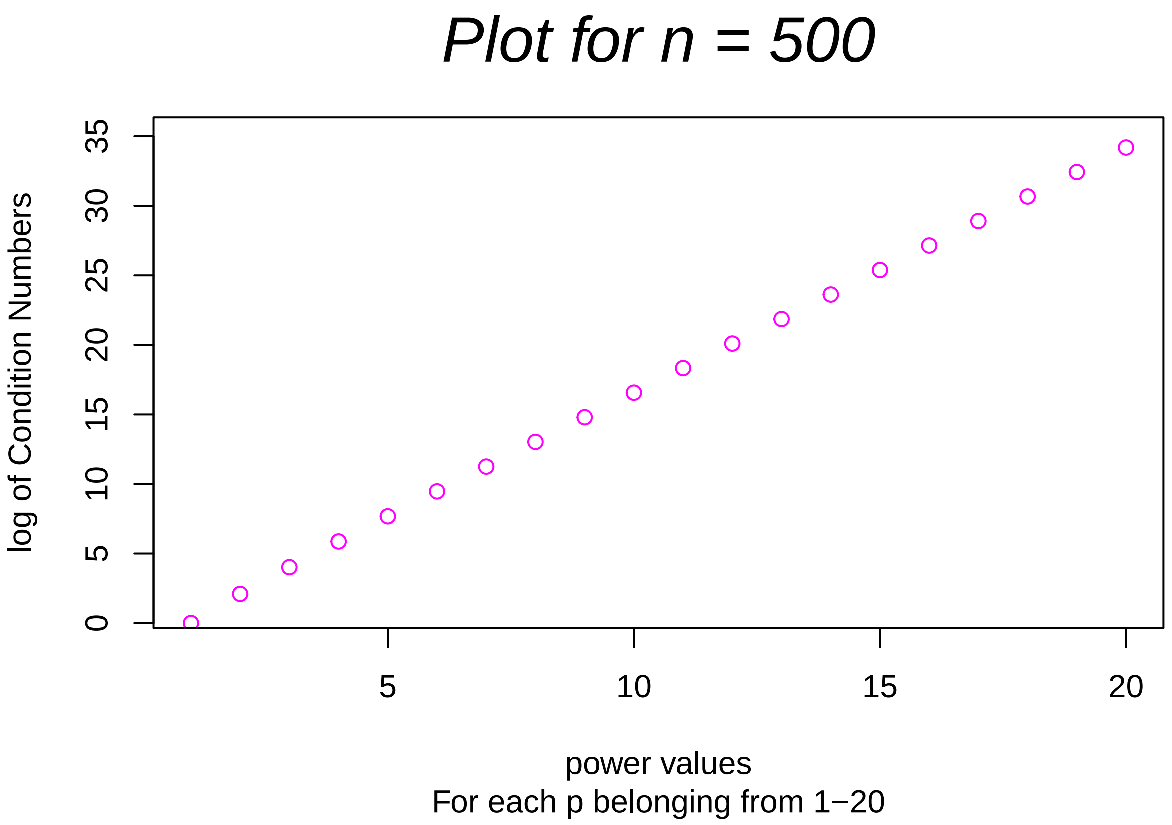
```r
condition_number(100, "cyan")
```



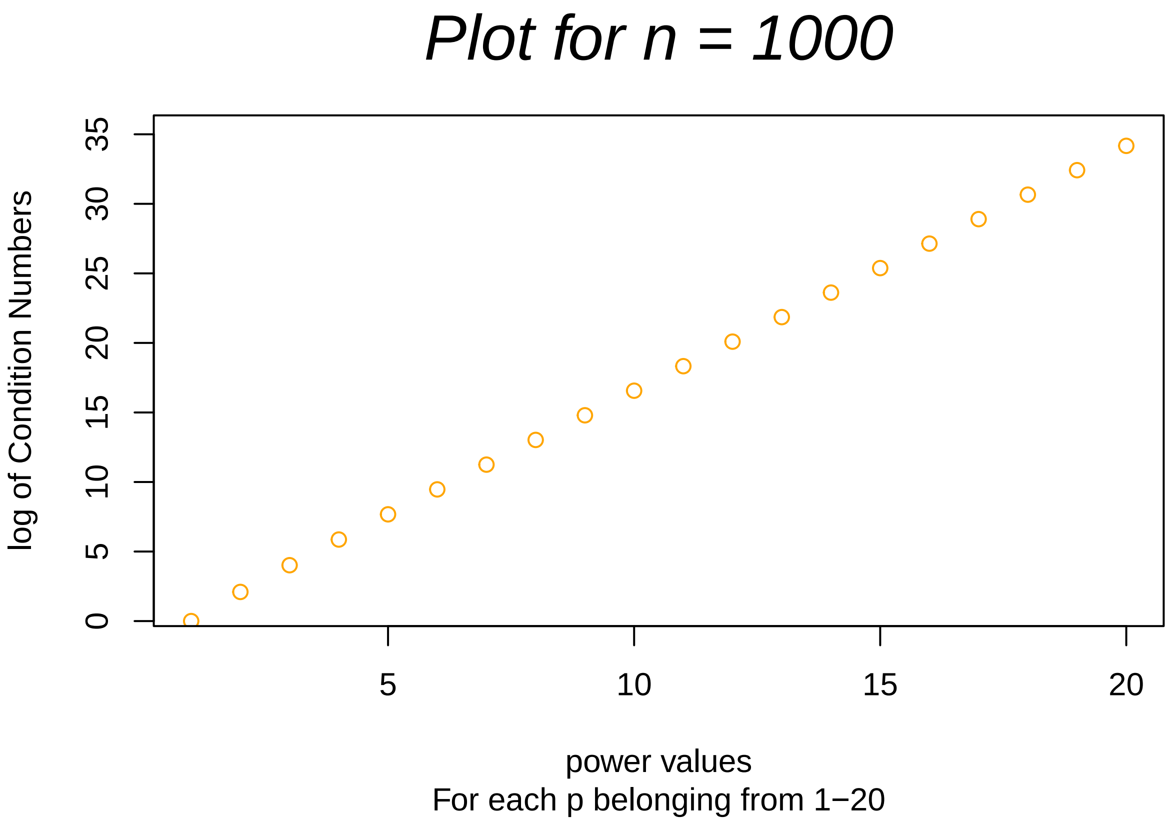
```r
condition_number(200, "red")
```



```r
condition_number(500, "magenta")
```



```
condition_number(1000, "orange")
```



Inference of question 1

As we can see from the graphs plotted above, for every p belong from 1 to 20 and for all xi = i/(n + 1) for i = 1,. . . .,n where n is ranging {30,50,100,200,500,1000}, we can see that the pattern is the same for all.

The condition number of the design matrix is an exponential function of degree p of the polynomial. As the graph shows for different values of n (number of data points), the condition number more or less remains the same for same value of p. That is for same value of p but for different n, the values are same.

It is to be noted that the condition number and the number of data points depends on many factors, such as the distribution of the data, the choice of basis functions, and the scaling of the design matrix The higher the degree of the polynomial, the more parameters there are to fit, making the design matrix larger and more sensitive to perturbations

Question 2

Piecewise constant fit

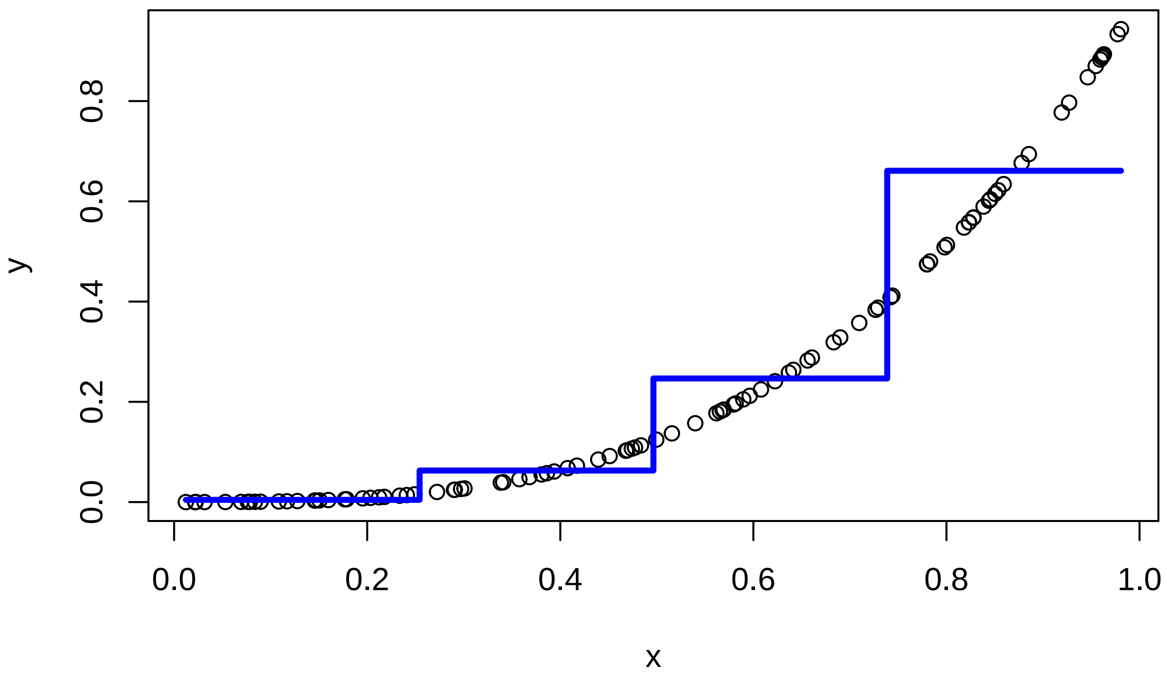
```
#Creating Piecewise constant funcyion
piecewiseConstant = function(x,y, L, plot=TRUE){
  n = 2^L
  #K = quantile(x, seq(0, 1, len = n+1), type=1)             # for quantile
  quotient = (range(x)[2]-range(x)[1])/n
  K = seq(from = range(x)[1], to = range(x)[2], by =quotient ) # for equal split
  pts = rep(0,2*n)
  val = rep(0,2*n)
  # looping from 1 to n
  for (j in 1:n){
    I = (K[j] < x)&(x <= K[j+1])
    if (length(I[I==TRUE]) !=0){
      fit = lm(y[I] ~ 1)
      pts[2*j-1] = K[j]
      pts[2*j] = K[j+1]
      val[2*j-1] = coef(fit)
      val[2*j] = coef(fit)
    }
    else{
      pts[2*j-1] = K[j]
      pts[2*j] = K[j+1]
      val[2*j-1] = val[2*j-3]
      val[2*j] = val[2*j-2]
    }

  }
  if (plot){
    if (L==2){
      lines(pts, val, col="blue", lwd = 3)
    }
    else if (L==3){
      lines(pts, val, col="green", lwd = 3)
    }
    if (L==4){
      lines(pts, val, col="red", lwd = 3)
    }
  }
}
```

2(a)

Write a function piecewiseConstant(x, y, L, plot = TRUE) taking in a one dimensional predictor variable x with values in [0, 1] and a response y, and fits a piecewise constant model (by least squares) on 2L intervals of equal length partitioning the unit interval (L is a nonnegative integer) in the form of a numerical vector of length 2L, with the option of producing a scatterplot with the fit overlaid.

```
# For part 2a
x = runif(100, min=0, max=1)
y = x^3
plot(y~x)
piecewiseConstant(x,y,2, TRUE)
```



Inference of 2a

As we fit piecewise constant model , we have different regression models for each of the intervals. The analysis for a particular interval can be done through that regression model, instead of using one model for the whole dataset points.

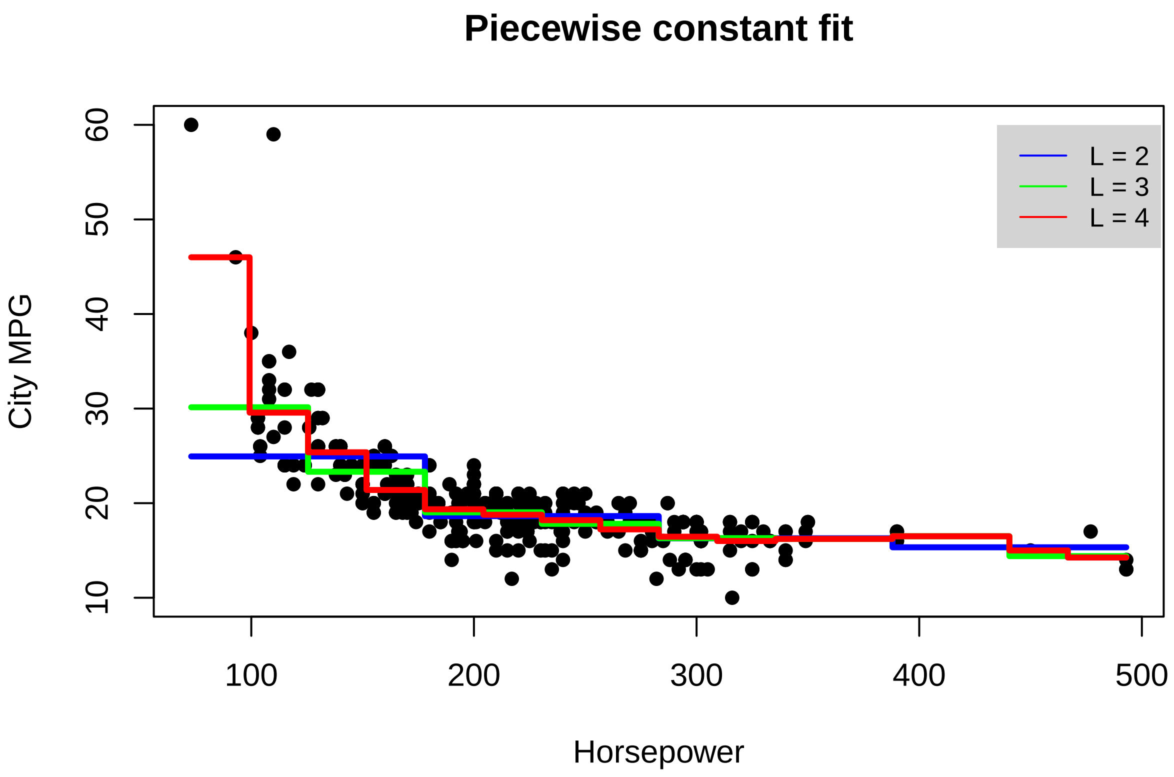
Thus piecewise approximation of a function or data points means each piece corresponds to a constant value over a particular interval.

2(b)

Apply your function to explaining City Mpg as a piecewise constant function of Horsepower in the 04cars dataset. Produce a single scatterplot, with lines corresponding to the fit with L = 2 (blue), L = 3 (green), and L = 4 (red). Add a legend, etc, so it looks 'nice'.

```
#Loading the cars dataset
#setwd("D:/projects/Quarter-2/Stats_model")#
load("../cars/04cars.rda")

data_ = dat
dat = data_[complete.cases(data_),]
#plotting for different values of L
plot(dat$Horsepower,dat$City_MPG, pch = 16, main="Piecewise constant fit",
     cex = 1, xlab="Horsepower", ylab="City MPG")
piecewiseConstant(dat$Horsepower,dat$City_MPG,2, TRUE)
piecewiseConstant(dat$Horsepower,dat$City_MPG,3, TRUE)
piecewiseConstant(dat$Horsepower,dat$City_MPG,4, TRUE)
legend(435, 60, legend=c("L = 2", "L = 3", "L = 4"),
       col=c("blue", "green", "red"),
       lty=1, cex=0.8, box.lty = 0, bg='lightgrey')
```



Inference of Question 2b

Note: for intervals with no data points, we have interpolated the previous regression line.

We see that as the number of partitioning intervals increase, we have more number of regressions for smaller intervals which increases the fit of the model to the data.As the intervals increase so does the variance and the bias decreases.

With more intervals, the model can capture more local variations in the data and better approximate the true function and hence reduced bias.

With more intervals, the model can over-fit the data and fit to the noise in the data, resulting in a higher variance in the predictions.

Team Contributions

Both the team members Sourabh Prakash and Priyanshi Shah have contributed equally to the homework by discussing the key points and logic together and doing pair programming. For the implementation part question 1 was contributed by Priyanshi Shah and question 2 by Sourabh Prakash. The infereneces were drawn together by both the team members.