HW 8 script

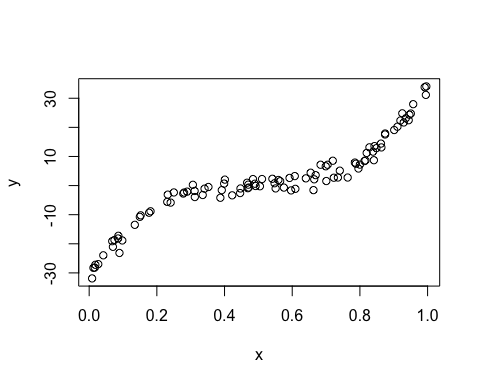
Ian Douglas

4/16/2019

# Question 1

### Part A

set.seed(26)  
x = runif(100, 0, 1)  
y = x + x^2 + 4\*(4\*x-2)^3 + 0.1\*exp(x) + rnorm(100,0,2)  
plot(x,y)

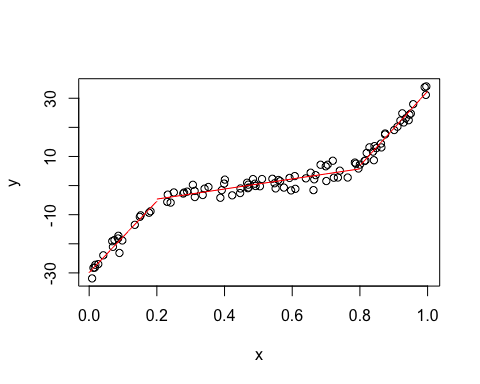
 ###Part B

cube.mod <- lm(y ~ poly(x, 3))  
RMSE <- sqrt(mean((predict(cube.mod) - y)^2))  
RMSE

## [1] 1.802683

### Part C

cats <- cut(x,breaks = c(0, 0.2, 0.8, 1))  
data <- as.data.frame(list(x = x, y = y,x.groups = cats))  
stp1 <- lm(y ~ x, data = data[data$x.groups == levels(cats)[1],])  
stp2 <- lm(y ~ x, data = data[data$x.groups == levels(cats)[2],])  
stp3 <- lm(y ~ x, data = data[data$x.groups == levels(cats)[3],])



RMSE1

## [1] 1.660987

RMSE2

## [1] 1.946148

RMSE3

## [1] 1.594409

### Part D

#repeat with two quadratic functions  
cats2 <- cut(x,breaks = c(0, 0.5, 1))  
data$x.groups2 <- cats2  
quad.stp1 <- lm(y ~ poly(x,4), data = data[data$x.groups2 == levels(cats2)[1],])  
quad.stp2 <- lm(y ~ poly(x,4), data = data[data$x.groups2 == levels(cats2)[2],])  
  
RMSE.quad1 <- sqrt(mean((predict(quad.stp1) - data[data$x.groups2 == levels(cats2)[1],2])^2))  
RMSE.quad2 <- sqrt(mean((predict(quad.stp2) - data[data$x.groups2 == levels(cats2)[2],2])^2))  
  
RMSE.quad1

## [1] 1.524457

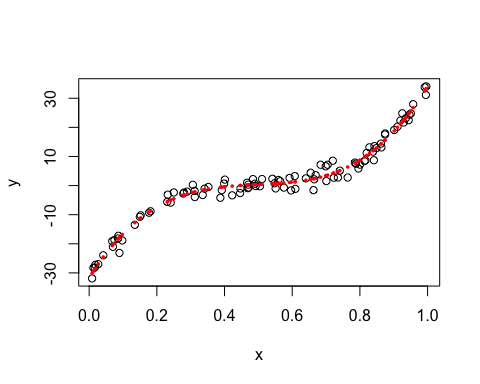
RMSE.quad2

## [1] 1.935115

### Part E

#Spline Regression  
library(splines)  
knots <- quantile(data$x, p = c(0.25, 0.5, 0.75))  
spline.mod <- lm(y ~ bs(x, knots = knots), data = data)  
RMSE.spline <- sqrt(mean((predict(spline.mod) - y)^2))  
RMSE.spline

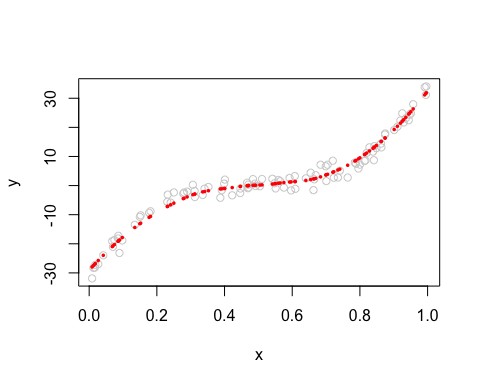
## [1] 1.801841



### Part F

#default loess curve  
loess.mod <- loess(y ~ x, data = data)  
RMSE.lo <- sqrt(mean((predict(loess.mod) - y)^2))  
RMSE.lo

## [1] 2.008462



### Part G

#Comparison and discussion  
data.frame(list(RMSE.cubic = RMSE,  
 RMSE.linear.stepfun = mean(c(RMSE1,RMSE2,RMSE3)),  
 RMSE.quadratic = mean(c(RMSE.quad1,RMSE.quad2)),  
 RMSE.loess = RMSE.lo,  
 RMSE.spline = RMSE.spline))

## RMSE.cubic RMSE.linear.stepfun RMSE.quadratic RMSE.loess RMSE.spline  
## 1 1.802683 1.733848 1.729786 2.008462 1.801841

Conclusion: The lowest RMSE was achieved by creating step functions, with partitions chosen at theoretically meaningful locations. Notably, this achieved the lowest overall RMSE for the entire domain of X, however, only on average, with some pieces of the step function fitting much more poorly than the spline and loess methods. Interestingly, the cubic polynomial did not achieve the lowest RMSE, suggeseting that even minimizing prediction error can in effect misguide an understanding of the true nature of the underlying function, due to the noise or error of the observed data. In that vein, the spline method may be considered the best approach, in that it most closely approximated the cubic model (in terms of RMSE), and thus may have struck the optimal balance between overfitting and underfitting.

# Question 2

### Part A

#read in data  
infrared =  
 read.table("http://www.stat.colostate.edu/computationalstatistics/datasets/infrared.dat",header=T)

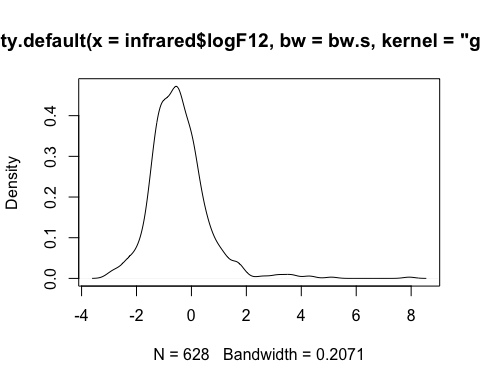
#### Estimate Silverman’s rule bandwith:

infrared$logF12 <- log(infrared$F12)  
IQR = IQR(infrared$logF12)  
min = min(c(sqrt(var(infrared$logF12)), (IQR/1.34)))  
n = length(infrared$logF12)  
  
bw.s <-.9\*min\*(n^(-1/5))  
bw.s

## [1] 0.2071217

### Part B

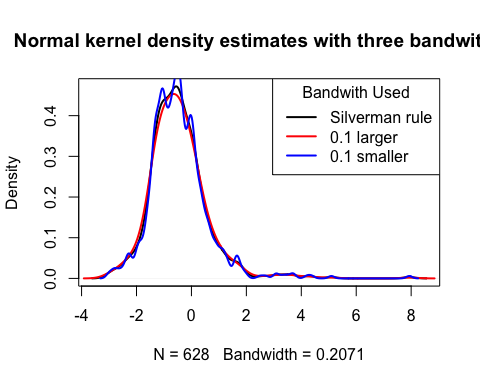
#use normal kernel and the above bandwith to obtain density estimates  
plot(density(infrared$logF12,bw=bw.s, kernel = "gaussian"))



### Part C

#### Repeat with one bw larger, and one smaller

plot(density(infrared$logF12,bw=bw.s, kernel = "gaussian"), col = 1, lwd = 2,  
 "Normal kernel density estimates with three bandwiths")  
lines(density(infrared$logF12,bw= (bw.s + .1), kernel = "gaussian"), col = 2,lwd = 2)  
lines(density(infrared$logF12,bw= (bw.s - .1), kernel = "gaussian"), col = 4,lwd = 2)  
legend(x = "topright", legend = c("Silverman rule","0.1 larger", "0.1 smaller"), col = c(1,2,4),lwd = 2, title = "Bandwith Used")



Discussion: The Silverman rule of thumb increased by a constant of .1 provided the smoothest curve, though it is unknown if it missed some detail in the density that the regular Silverman rule of thumb provided (showing almost a local maximum just to the left of the global maximum). The blue line, which was produced with a bandwidth smaller than .1 than the Silverman rule created many irregularities. Nevertheless, if the goal were to identify a global maximum, this very erratic line may have been preferred, since it showed the largest density around the global maximum (whereas the other two may have underestimated that point).

### Part D

h <- 2.15\*sqrt(var(infrared$logF12))\*(n^(-1/5))  
logF12 <- infrared$logF12  
br <- pretty(logF12, diff(range(logF12)) / h)  
brplus <- c(min(br)-h, max(br+h))  
histg <- hist(logF12, breaks = br, freq = FALSE,  
 main = "", xlim = brplus)  
  
vx <- histg$mids  
vy <- histg$density  
delta <- diff(vx)[1]  
k <- length(vx)  
vx <- vx + delta  
vx <- c(vx[1] - 2 \* delta, vx[1] - delta, vx)  
vy <- c(0, vy, 0)  
  
# add the polygon to the histogram  
polygon(vx, vy)

