Homework 5

Ben Buzzee

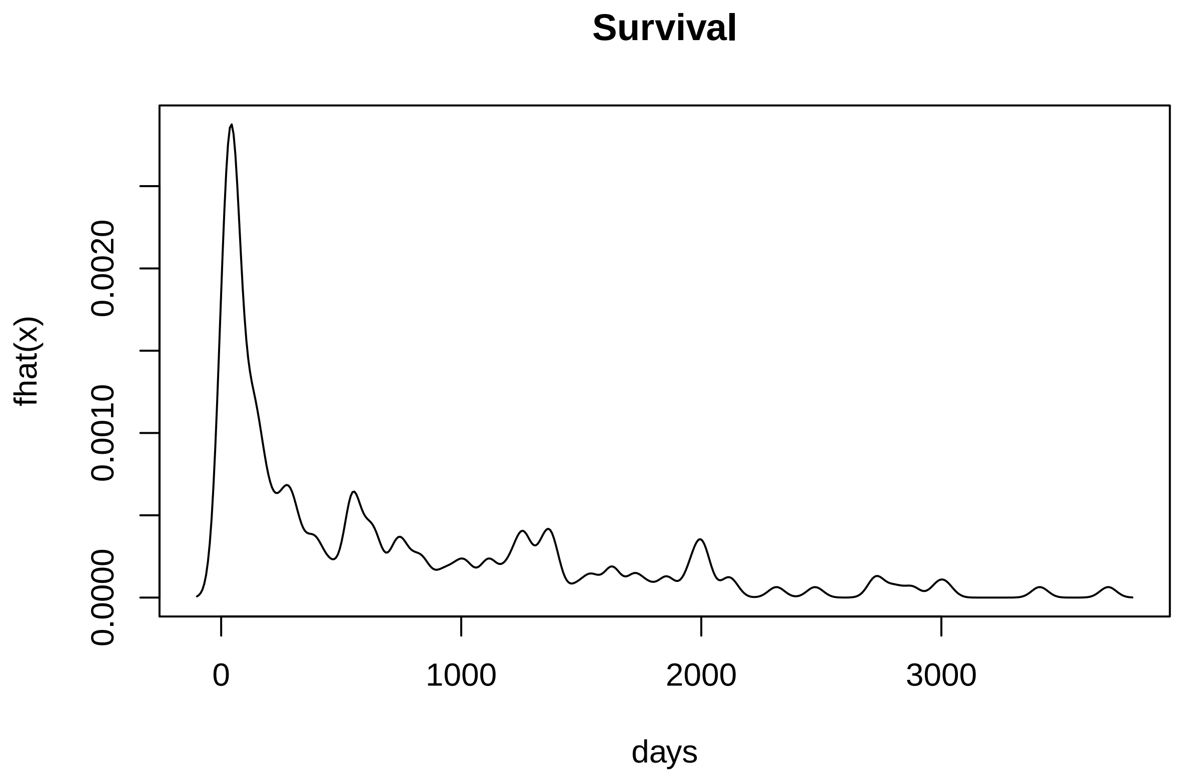
October 22, 2019

1. Density Estimation

(a)

We will use a gaussian kernel and choose a bandwidth using unbiased cross validation.

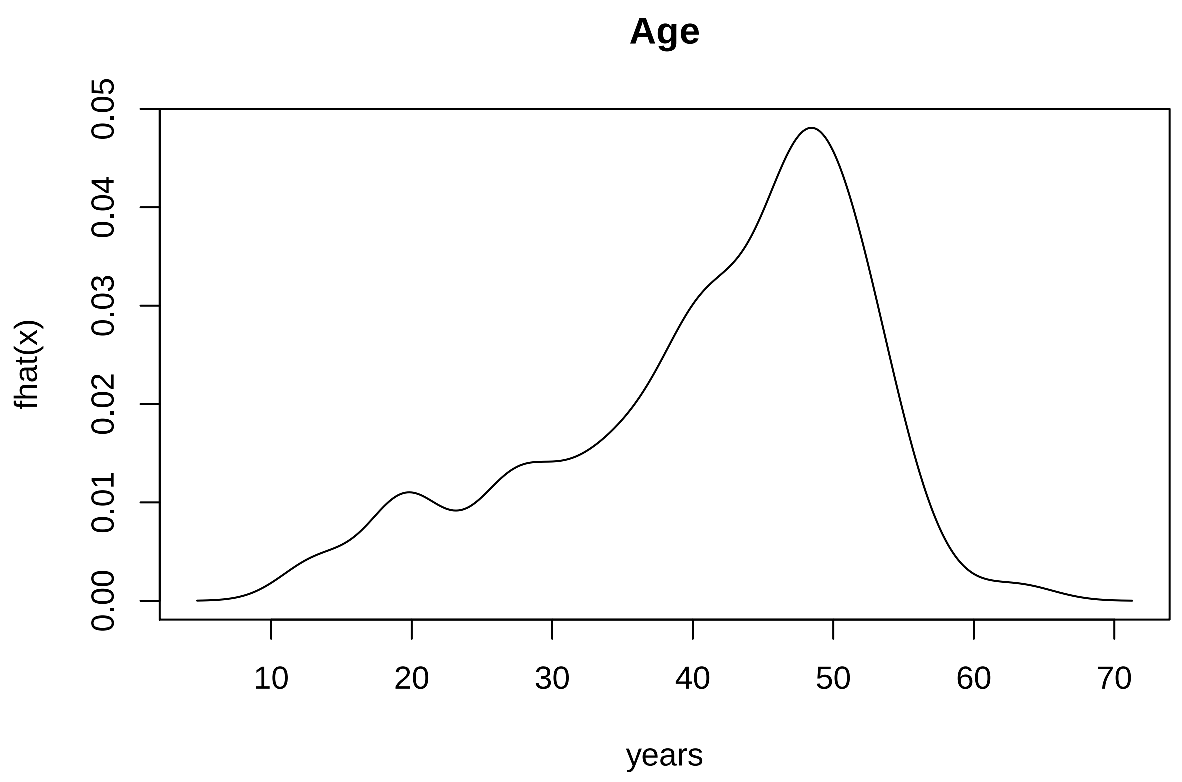
```r
heart <- read.csv("heart.txt", sep = " ")

fhat1.gauss <- density(heart$surv, kernel="gaussian", bw="ucv")
plot(fhat1.gauss$x, fhat1.gauss$y, type='l', xlab='days', ylab='fhat(x)', main = "Survival")
```



We see from the above plot that survival times have a strong right skew, and the probability of a death is highest in the first 500 days.

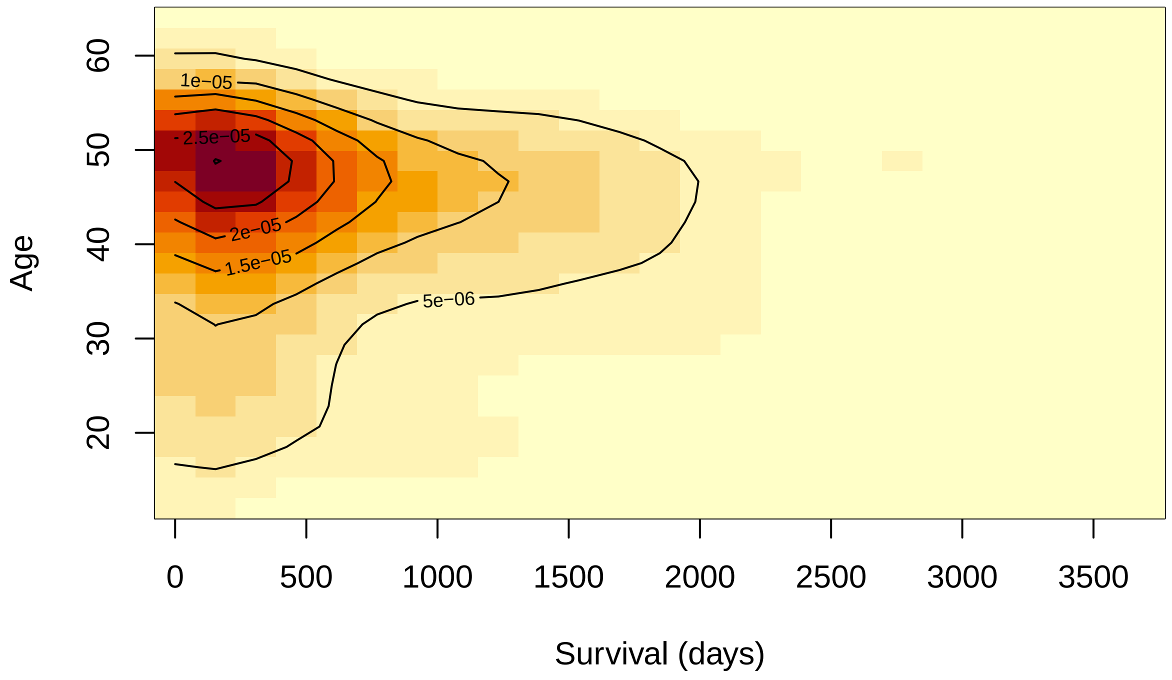
(b)

```r
fhat1.gauss <- density(heart$age, kernel="gaussian", bw="ucv")
plot(fhat1.gauss$x, fhat1.gauss$y, type='l', xlab='years', ylab='fhat(x)', main = "Age")
```



The density function for age is unimodal and skewed to the left. Subjects are much more likely to be 30-60 than any other age. Ages could potentially be right-censored after age 65.

(c)



From the contour plot we can see that age appears to be a factor in determining the probability of survival. The probability of death is much higher in the first 500 days for those in the 40-55 age group.

2. MISE

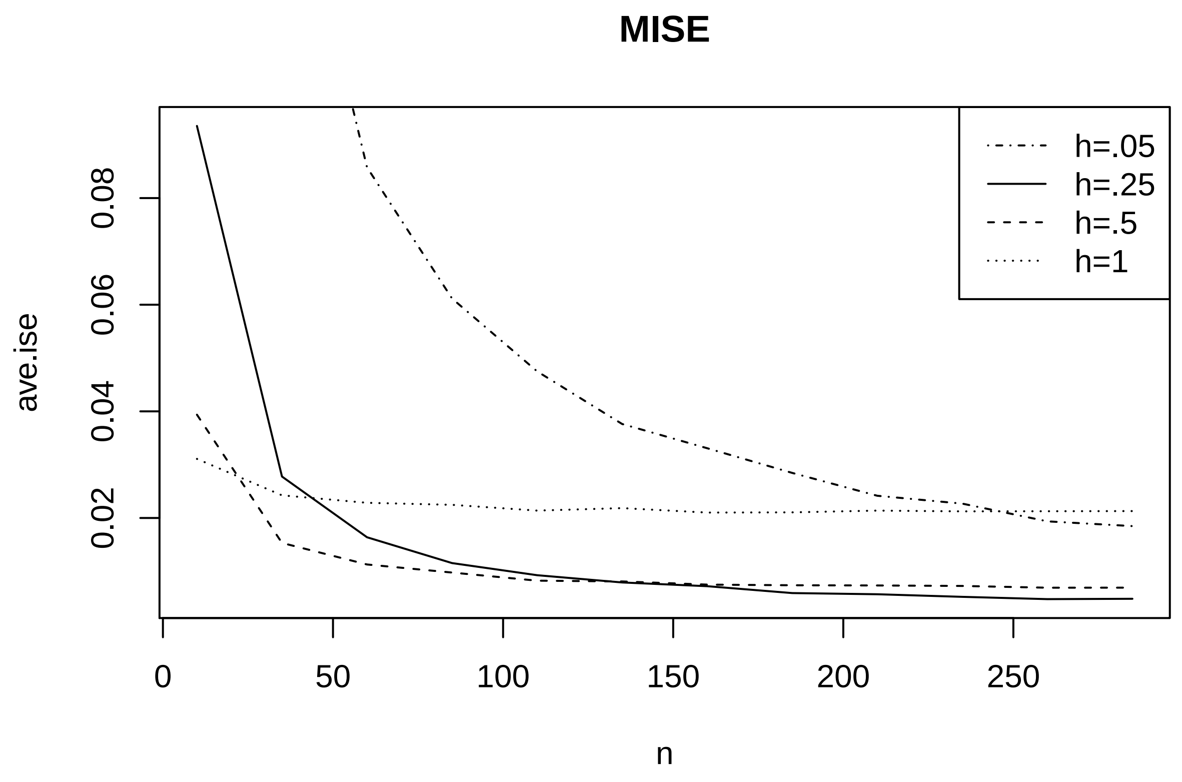
(a)

The MISE is composed of two terms. As lambda varies from very small to very large, one of the terms shifts from very large to very small. For very small λ, $\frac{1}{n\lambda}\int K^2(x)dx$ becomes very large and $\frac{\lambda^4}{4}\mu_2\int f''(x)^2dx$ becomes very small. The opposite happens when λ becomes very large. So we need to find a balance between those two terms in order to minimize MISE.

(b)

```
ise <- function(ftrue,fhat,xgrid){
  w=xgrid[2]-xgrid[1]
  d=fhat-ftrue
  y=w*t(d)%*%d
  return(y)
}
```

```r
mise.gam=function(n, alpha, beta, xgrid, B, lambda){

  out <- rep(NA,length(n))
  # sample from gamma distribution
  f <- dgamma(xgrid, shape = alpha, scale = beta)

  for(i in 1:length(n)){

    temp=rep(NA, B)

    for(j in 1:B){

    x=rgamma(n[i], shape = alpha, scale = beta)
    fhat=density(x, from = min(xgrid), to = max(xgrid), n = length(xgrid), bw = lambda)
    temp[j]=ise(f,fhat$y,xgrid)
    }

  out[i]=mean(temp)
  }
return(out)
}

n <- seq(from = 10, to = 300, by = 25)
xgrid <- seq(from = 0, to = 10, by = .2)
lambda <- seq(from = 0, to = 10, by = .2)


ave.ise.05 <- mise.gam(n, alpha = 2, beta = 1,xgrid = xgrid, B = 250, lambda = .05)
ave.ise.25 <- mise.gam(n, alpha = 2, beta = 1,xgrid = xgrid, B = 250, lambda = .25)
ave.ise.5 <- mise.gam(n, alpha = 2, beta = 1,xgrid = xgrid, B = 250, lambda = .5)
ave.ise1 <- mise.gam(n, alpha = 2, beta = 1,xgrid = xgrid, B = 250, lambda = 1)


par(mfrow=c(1,1))
plot(n,ave.ise.25,type='l',ylab='ave.ise', main = "MISE")
lines(n,ave.ise.5,lty=2)
lines(n,ave.ise1,lty=3)
lines(n, ave.ise.05, lty = 4)
legend("topright", c("h=.05","h=.25","h=.5", "h=1"),lty=c(4,1,2,3))
```

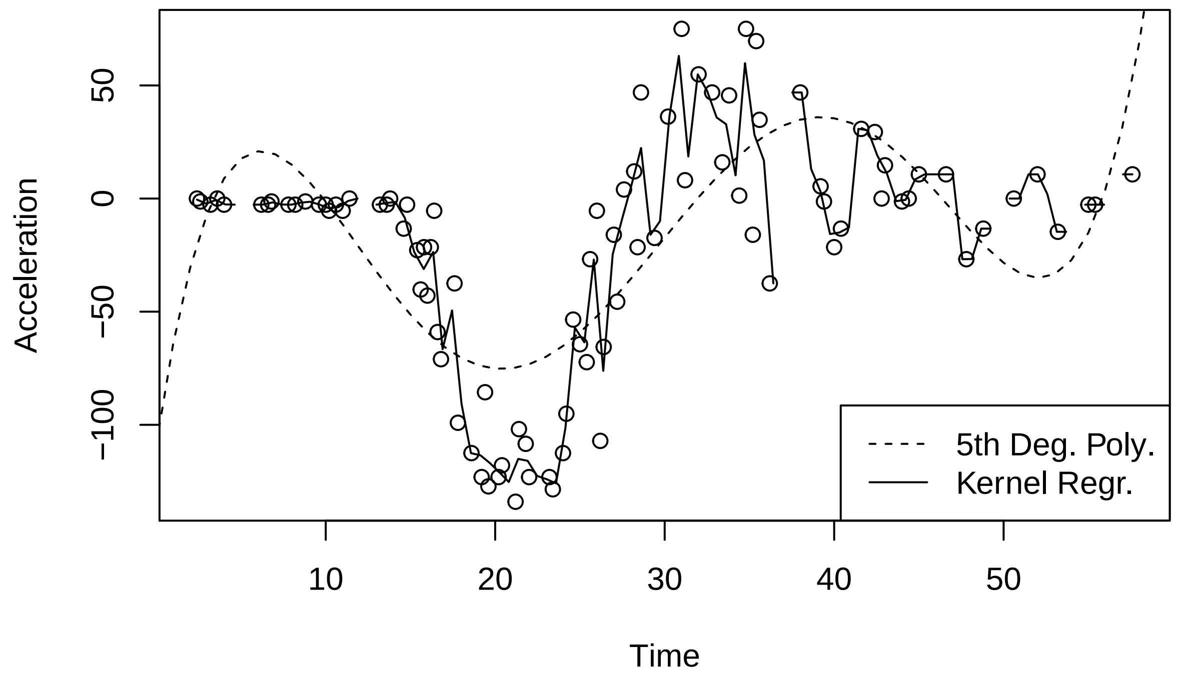


We see that MISE is minimized somewhere between h = .25 and h = .5. As a function of n the MISE levels out around the 150 mark.

3. Kernel Regression

(a and b)

```
motor <- read.csv("motor.txt", sep = "\t")

x <- motor$times
y <- motor$accel

plot(x,y, xlab = "Time", ylab = "Acceleration")
ghat=ksmooth(x, y, kernel="normal")
lines(ghat$x,ghat$y)


ghat.quad=lm(y~poly(x,5))
xvals=seq(0,60)
pred.quad=predict.lm(ghat.quad,data.frame(x=xvals))
lines(xvals,pred.quad,lty=2)
legend("bottomright", legend = c("5th Deg. Poly.", "Kernel Regr."), lty = 2:1)
```



The kernel regression does a much better job at "fitting"" the data. Although the polynomial doesn't fit as closely to the points it is probably more robust and possibly better at predicting future observations since it captures a more general trend.

4. Local Polynomial Regression

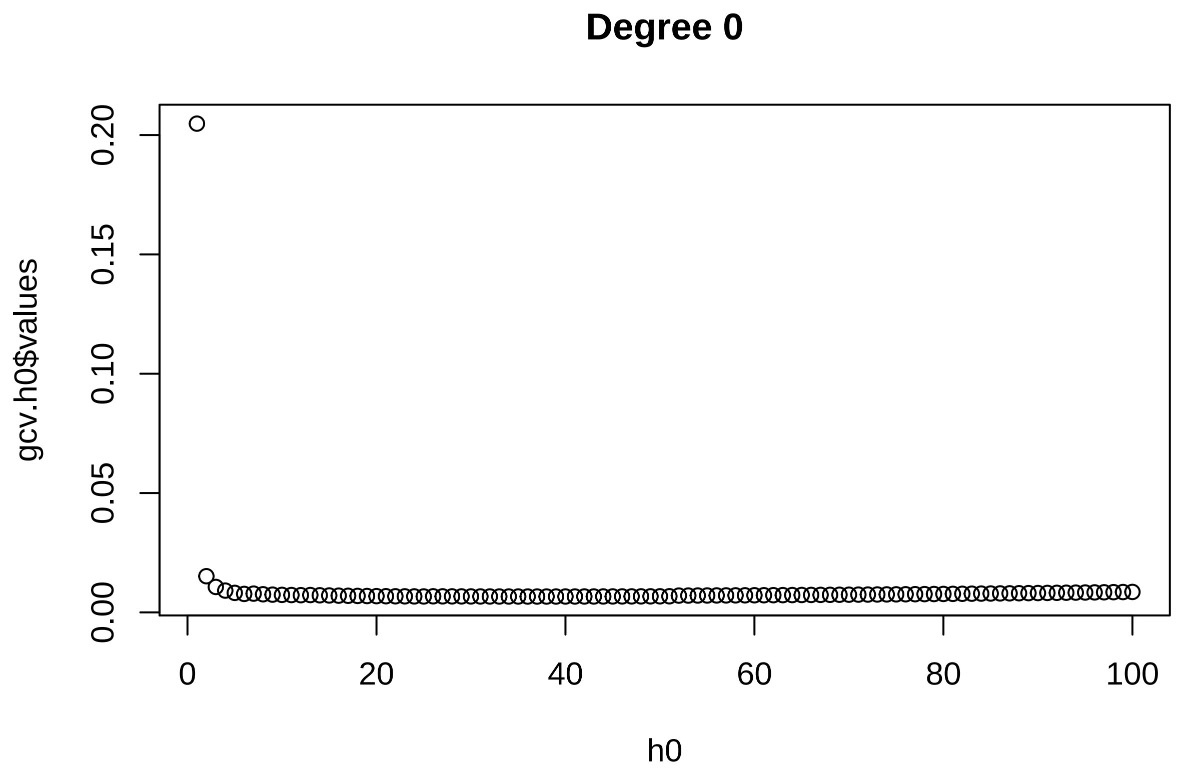
(a)

Our model will be $y_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_p x_i^p + e_i$ for each polynomial of order p. We are assuming our explanatory variables are known without error, that the true expected value of our variable can be expressed as a polynomial, and that we have a constant variance around this expected value.

(b through d)

```
library(locfit)
lidar <- read.csv("lidar.txt", sep = " ")




### degree 0 poly ###
h0=seq(1, 100, by=1)
alphamat= matrix(0, ncol=2, nrow=length(h0))
```

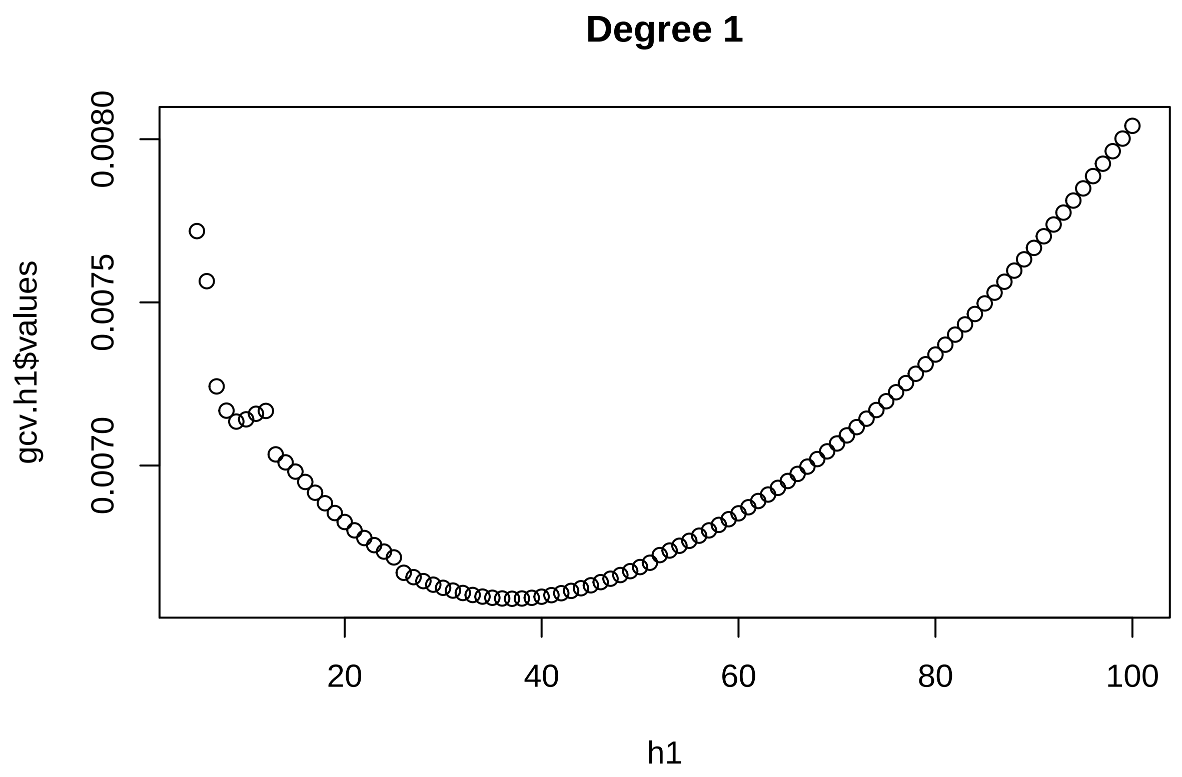
```r
alphamat[,2]=h0
gcv.h0 <- gcvplot(logratio~range, data = lidar, deg = 0, alpha = alphamat, maxk = 10000)
plot(h0, gcv.h0$values, main = "Degree 0")
```



```r
h0.opt=h0[gcv.h0$values==min(gcv.h0$values)]
h0.opt
```

```
## [1] 37
```

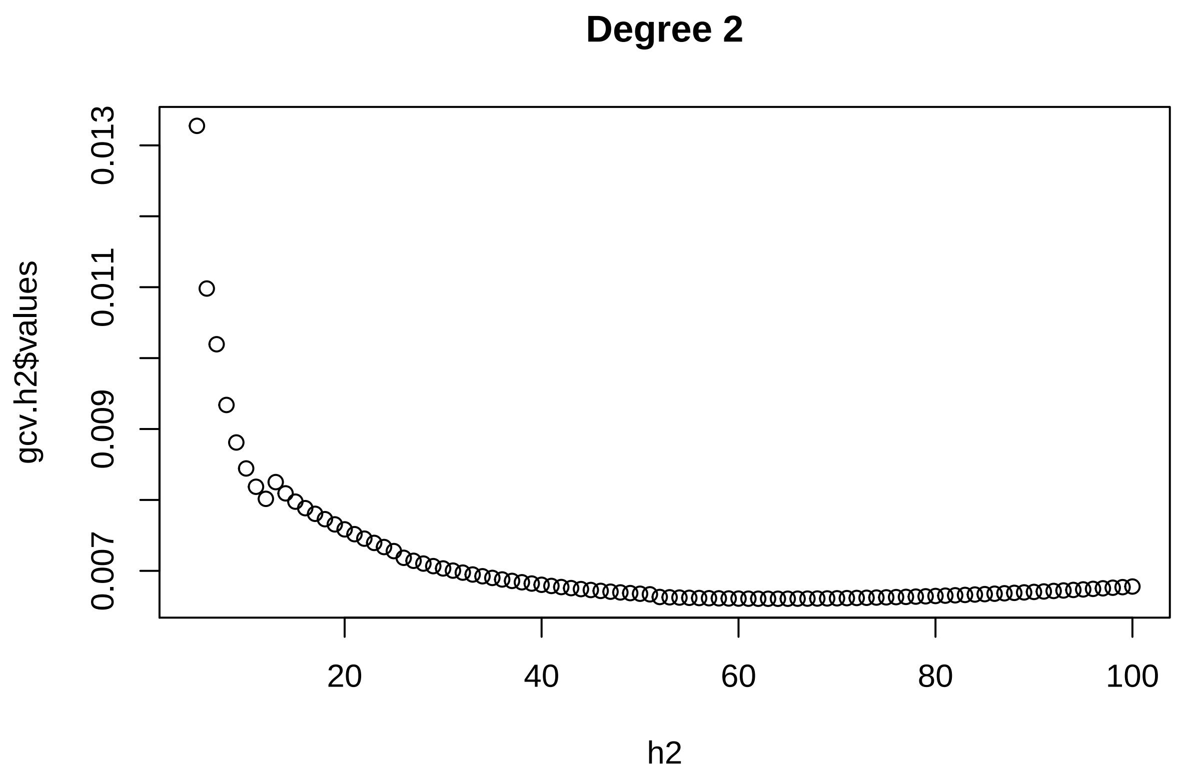
```r
# fit using optimum value
fit0=locfit(logratio~range, data=lidar, alpha = c(0, h0.opt))



### degree 1 poly ###
h1=seq(5, 100, by=1)
alphamat= matrix(0, ncol=2, nrow=length(h1))
alphamat[,2]=h1
gcv.h1 <- gcvplot(logratio~range, data = lidar, deg = 1, alpha = alphamat, maxk = 10000)
plot(h1, gcv.h1$values, main = "Degree 1")
```



```
h1.opt=h1[gcv.h1$values==min(gcv.h1$values)]
h1.opt
```

```
## [1] 37
```

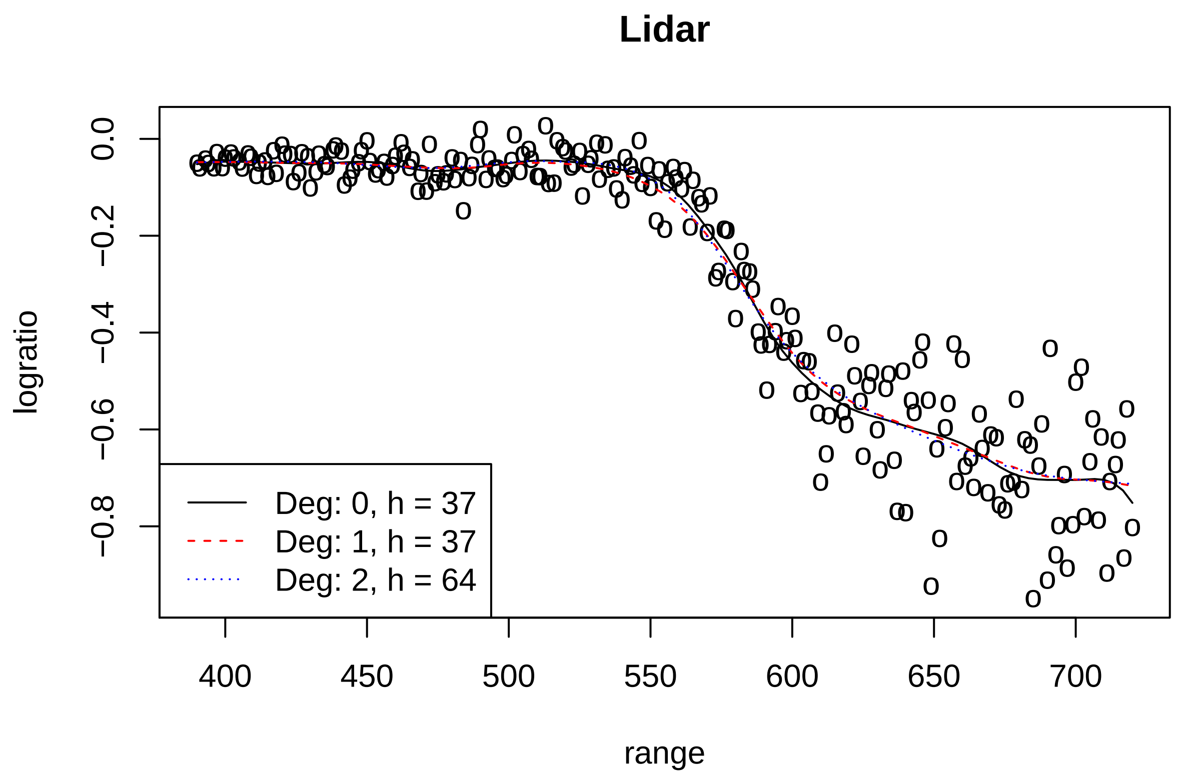
```
fit1=locfit(logratio~range, data=lidar, alpha = c(0, h1.opt), deg = 1)



### degree 2 poly ###
h2=seq(5, 100, by=1)
alphamat= matrix(0, ncol=2, nrow=length(h2))
alphamat[,2]=h2
gcv.h2 <- gcvplot(logratio~range, data = lidar, deg = 2, alpha = alphamat, maxk = 10000)
plot(h2, gcv.h2$values, main = "Degree 2")
```



```r
h2.opt=h2[gcv.h2$values==min(gcv.h2$values)]
h2.opt
```

```
## [1] 64
```

```r
fit2=locfit(logratio~range, data=lidar, alpha = c(0, h2.opt), deg = 2)





par(mfrow = c(1,1))
plot(fit0, get.data=T, main = "Lidar")
lines(fit1, lty = 2, col = "red")
lines(fit2, lty = 3, col = "blue")
legend("bottomleft",
       legend = c("Deg: 0, h = 37",
                  "Deg: 1, h = 37",
                  "Deg: 2, h = 64"),
       col = c("black", "red", "blue"),
       lty = 1:3)
```



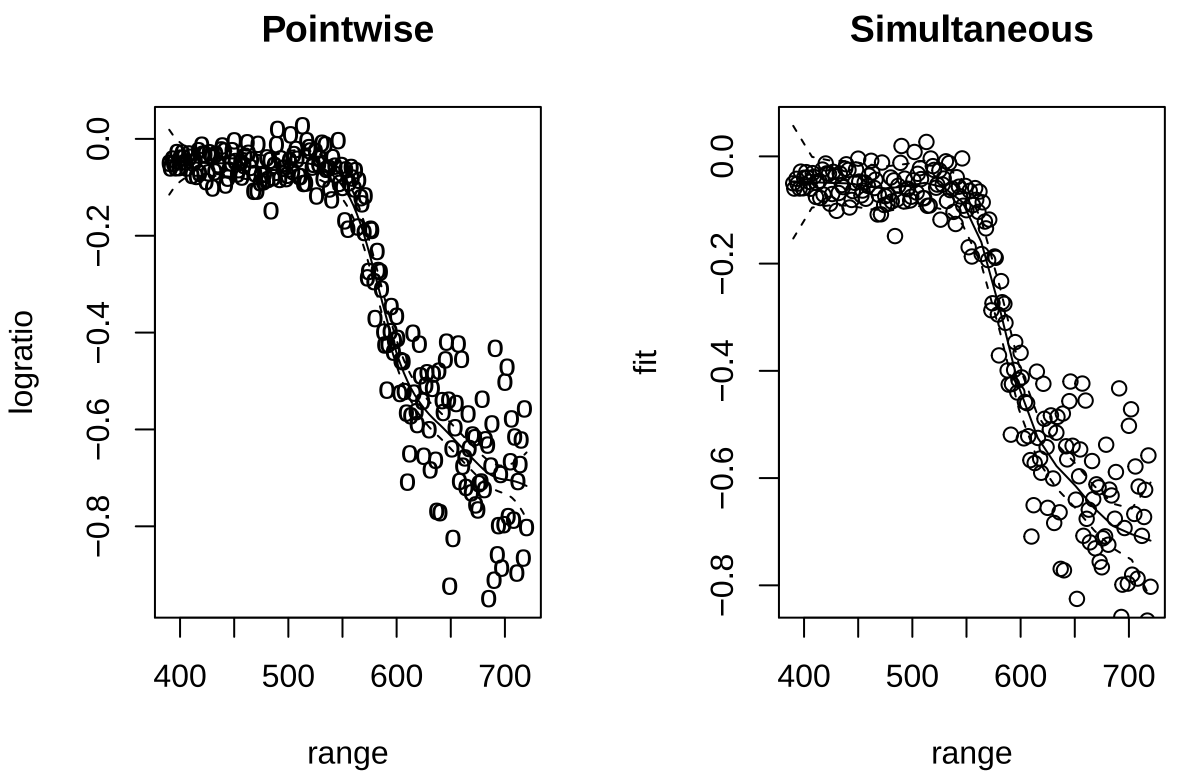
(e)

As seen above we arrived at very similar results. I did experience some difficulty choosing a range of h values to fit–getting the error "newsplit: out of vertex space" frequently. As such I'm skeptical the results are correct. In general the one and two degree local polynomials seem a bit more flexible, but for all practical purposes the result appears to be about the same.

5. Confidence Bands

We will use the linear smoother from above to create confidence bands:

```r
par(mfrow = c(1,2))
plot(fit1, get.data=T, band="global") #global specifies use variance est sigma.hat^2
title("Pointwise")

new.fit3=scb(logratio~range, deg=1, alpha=c(0,h1.opt), data=lidar)
plot(new.fit3)
title("Simultaneous")
```



If we were interested in a confidence interval for the function at a single x, then pointwise interval would be appropriate. Since we will probably be more interested in the function over it's entire domain, it would be better to draw conclusions based on the simultaneous interval. Using the simultaneous interval we could draw a conclusion about the true value of the function based on the confidence bands and be right about 95% of the time we make that inference.