Assignment 4: Conditional Variance and Local Poisson

Víctor Villegas, Roger Llorenç, Luis Sierra

2024-03-19

1. Conditional Variance

We are using *Aircraft data*, from the R library `sm`. These data record the following characteristics of aircraft designs.

- `Yr`
- `Period`
- `Power`
- `Span`
- `Length`
- `Weight`
- `Speed`
- `Range`

We begin by loading the library and transforming the data taking logs (except for `Yr` and `Period`).

```
# Clears plots
while (dev.cur() != 1) {
  dev.off()
}
# Clears global environment
rm(list=ls())


library(sm)
```

```
## Warning: package 'sm' was built under R version 4.3.3
```

```
## Package 'sm', version 2.2-6.0: type help(sm) for summary information
```

```
data(aircraft)
help(aircraft)
```

```
## starting httpd help server ...
```

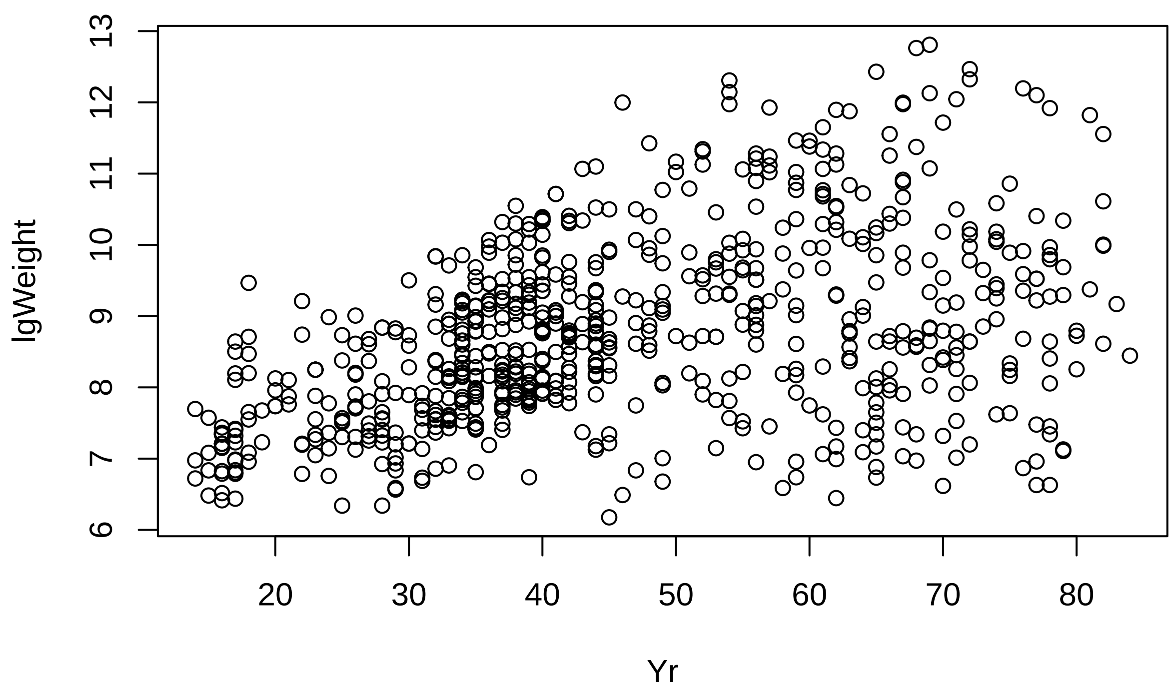
```
##  done
```

```
attach(aircraft)
lgPower <- log(Power)
lgSpan <- log(Span)
lgLength <- log(Length)
lgWeight <- log(Weight)
lgSpeed <- log(Speed)
lgRange <- log(Range)
```

We consider a heteroscedastic regression model $Y = m(X) + \sigma(X)\varepsilon$ for ε the standard, zero-mean Gaussian noise.

We are going to estimate the conditional variance of `lgWeight` (Y) given `Yr` (x). We can see the evolution of the (log) weight of the airships over the years in the following plot.

```
plot(Yr, lgWeight)
```



1.1. Nonparametric regression model on the original data

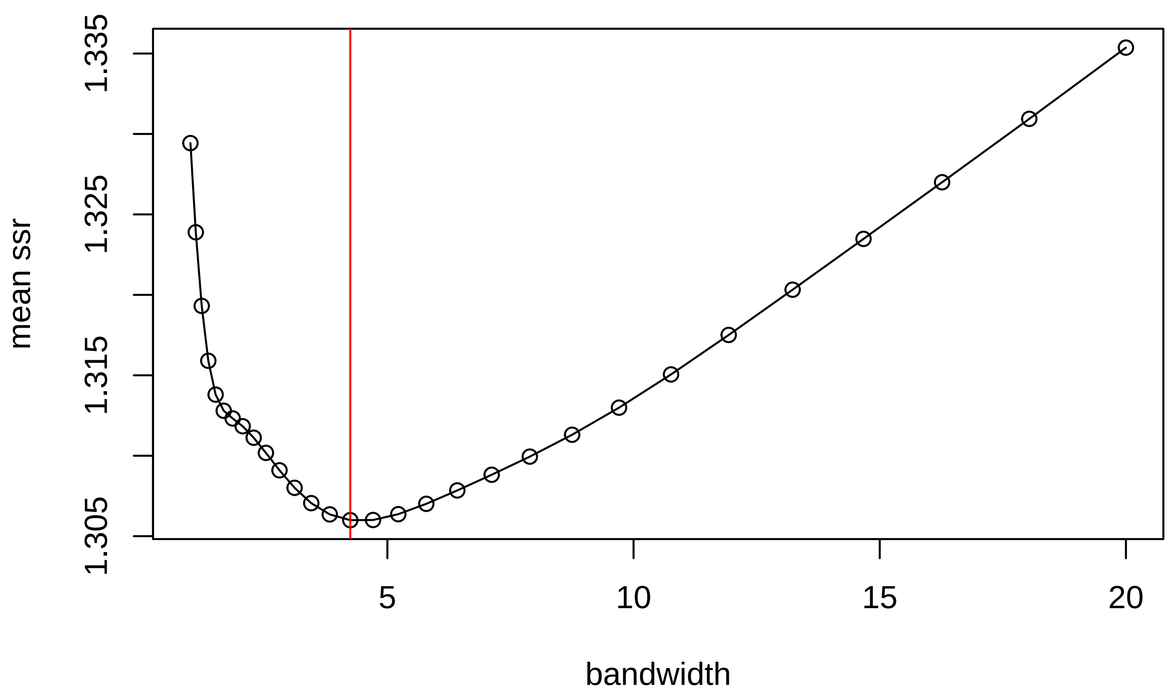
Option 1: using `loc.pol.reg`

```
# Function loc.pol.reg option
source("locpolreg.R")

# Leave-one-out CV to select bandwidth
h.cv.gcv <- function(x,y,h.v = exp(seq(log(diff(range(x))/20),
                                        log(diff(range(x))/4),l=10)),
                     p=1,type.kernel="normal"){
  n <- length(x)
  cv <- h.v*0
  gcv <- h.v*0
  for (i in (1:length(h.v))){
    h <- h.v[i]
    aux <- locpolreg(x=x,y=y,h=h,p=p,tg=x,
                     type.kernel=type.kernel, doing.plot=FALSE)
    S <- aux$S
```

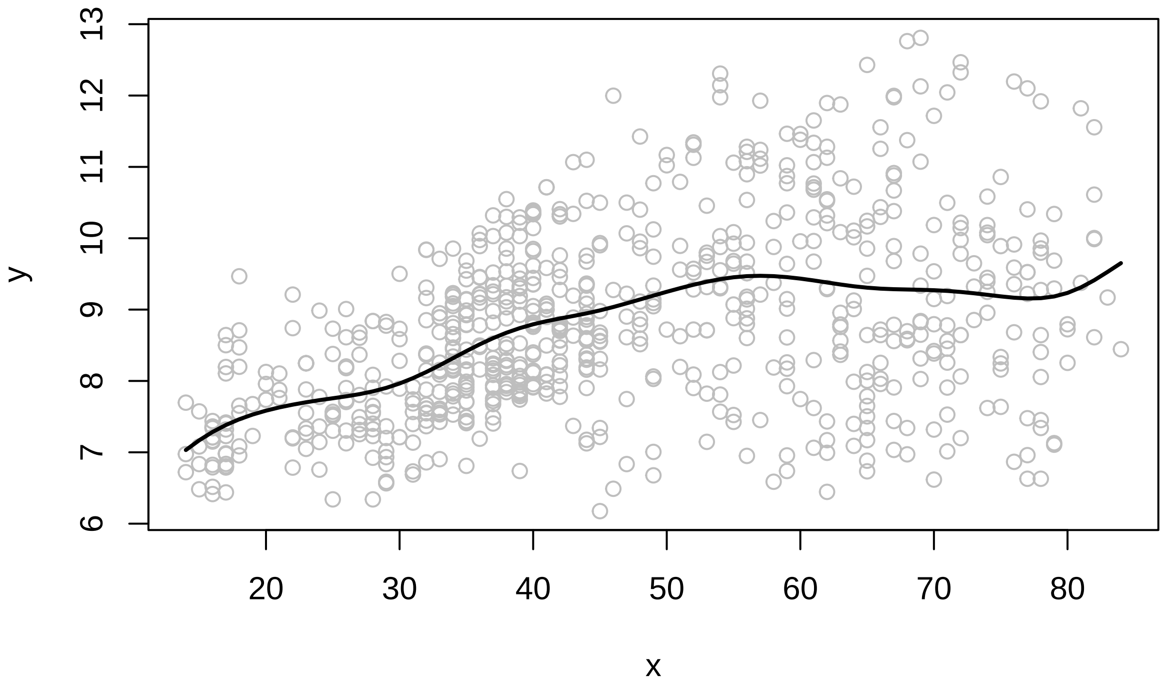
```r
    h.y <- aux$mtgr
    hii <- diag(S)
    av.hii <- mean(hii)
    cv[i] <- sum(((y-h.y)/(1-hii))^2)/n
    gcv[i] <- sum(((y-h.y)/(1-av.hii))^2)/n
  }
  return(list(h.v=h.v,cv=cv,gcv=gcv))
}
```

```r
h.v <-  exp(seq(from=log(1), to = log(20), length=30))
out.h.cv <- h.cv.gcv(x=aircraft$Yr, y=lgWeight, h.v=h.v)
h.loo.cv <- h.v[which.min(out.h.cv$cv)]
```

```r
plot(h.v,out.h.cv$cv, xlab ="bandwidth", ylab = "mean ssr")
lines(h.v,out.h.cv$cv)
abline(v = h.loo.cv, col = "red")
```




```r
aircraft.lp_reg <-locpolreg(x=aircraft$Yr,y=lgWeight,h=h.loo.cv)
```

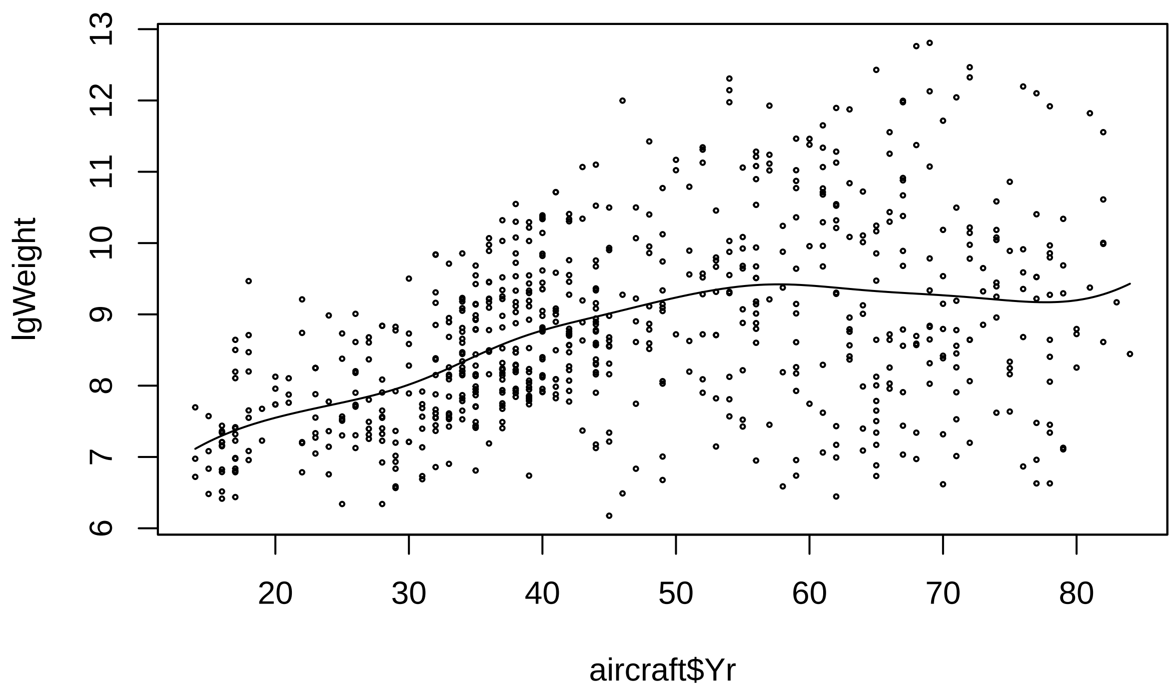


Option 2: using `sm.regression`

```
# Function sm.regression option
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
h2 <- dpill(x=aircraft$Yr,y=lgWeight,gridsize=length(aircraft$Yr),range.x=range(aircraft$Yr))
set.seed(123)
aircraft.sm_reg <- sm.regression(x = aircraft$Yr, y = lgWeight, h = h2, eval.points = seq(min(aircraft$Y
```

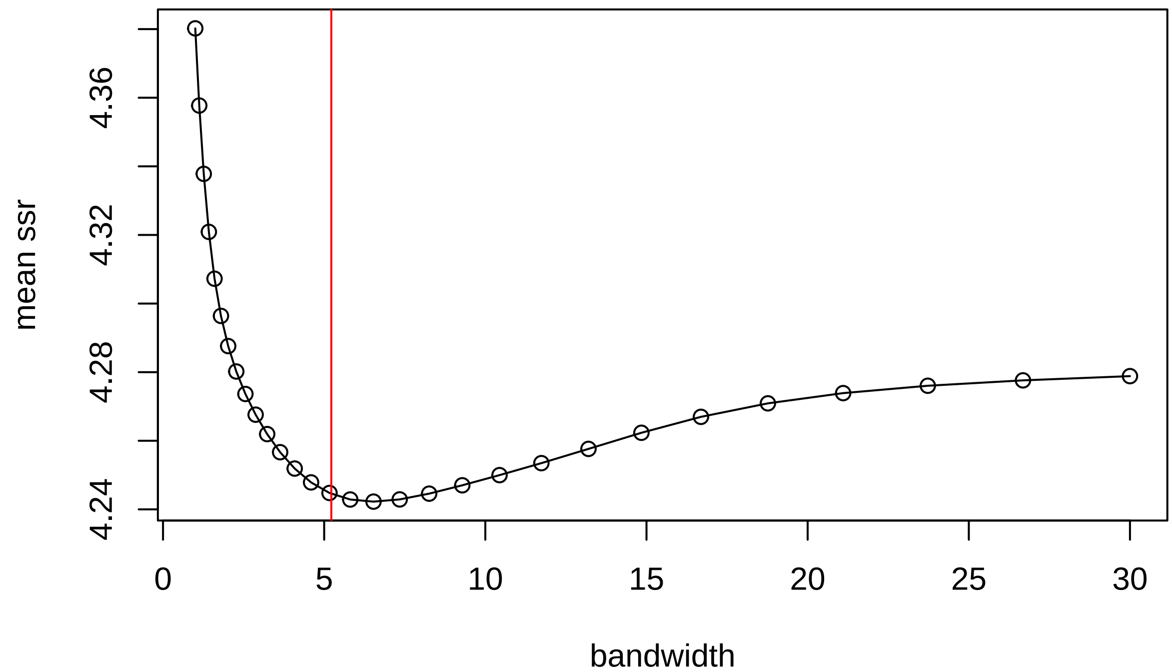


1.2. Transformed estimated residuals

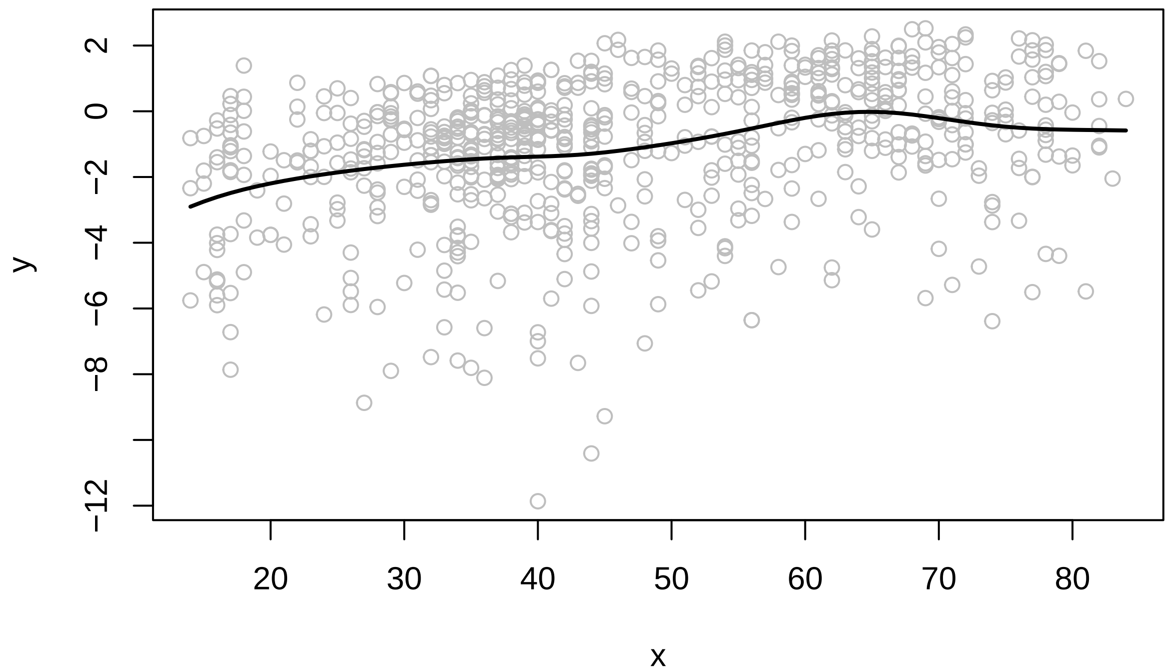
Option 1: using `loc.pol.reg`

```
# Function loc.pol.reg option
residualss1 <- lgWeight - aircraft.lp_reg$mtgr
z_i1 = log(residualss1^2)
h.v_z <-  exp(seq(from=log(1), to = log(30), length=30))

out.h.cv_z <- h.cv.gcv(x=aircraft$Yr, y=z_i1, h.v=h.v_z)
h.loo.cv_z <- h.v[which.min(out.h.cv_z$cv)]
```

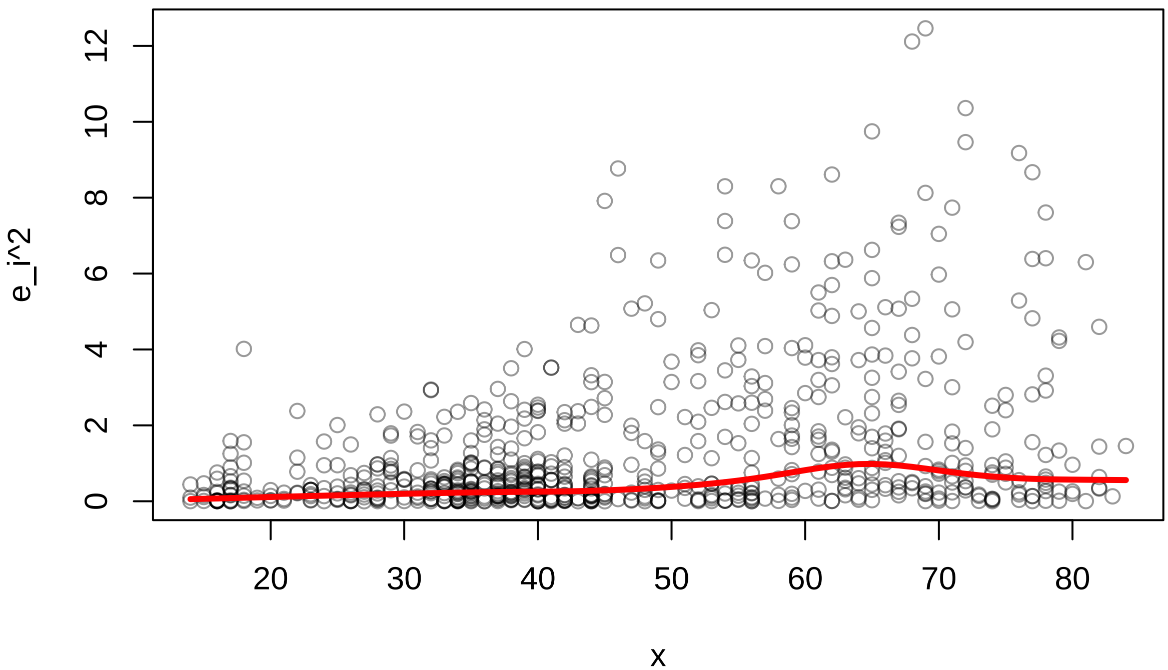
```
plot(h.v_z,out.h.cv_z$cv, xlab ="bandwidth", ylab = "mean ssr")
lines(h.v_z,out.h.cv_z$cv)
abline(v = h.loo.cv_z, col = "red")
```



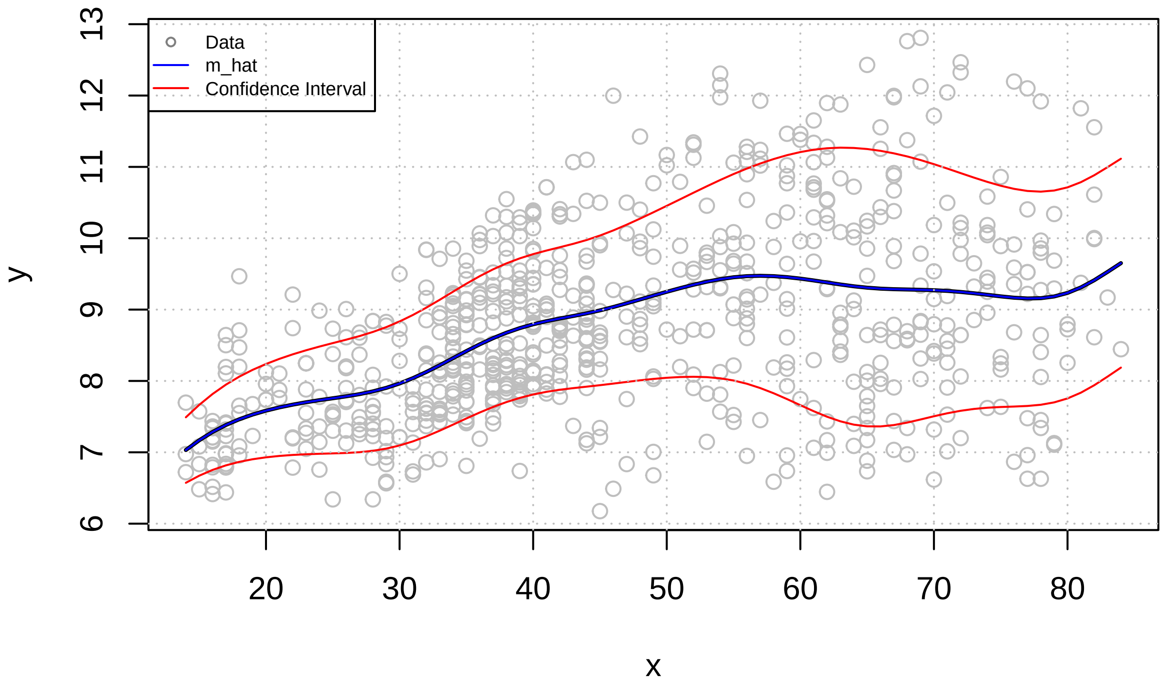

```
q_hat1 = locpolreg(x=aircraft$Yr,y = z_i1,h = h.loo.cv_z)
```




```
sigma_square_hat1 = exp(q_hat1$mtgr)


plot(aircraft$Yr, residualss1^2, col = rgb(0, 0, 0, alpha = 0.4), xlab = "x", ylab = "e_i^2")
lines(aircraft$Yr, sigma_square_hat1, col = "red", lwd = 3)  # Adjust line width here
```

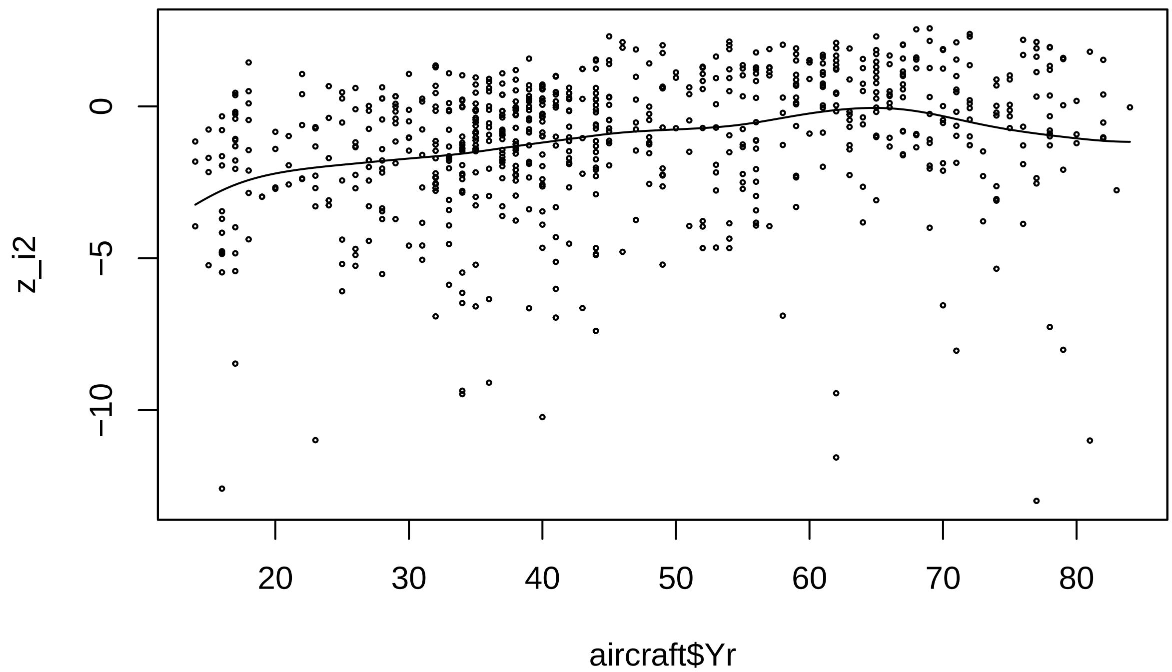



```
# Your plot code
plot_aux <- locpolreg(x=aircraft$Yr,y=lgWeight,h=h.loo.cv)
lines(aircraft$Yr, aircraft.lp_reg$mtgr, type = "l", col = "blue")
lines(aircraft$Yr, aircraft.lp_reg$mtgr + 1.96 * sqrt(sigma_square_hat1), col = "red")
lines(aircraft$Yr, aircraft.lp_reg$mtgr - 1.96 * sqrt(sigma_square_hat1), col = "red")
grid(col = "gray", lty = "dotted")  # Add grid

# Add legend with adjusted parameters
legend("topleft", legend = c("Data", "m_hat", "Confidence Interval"),
       col = c(rgb(0, 0, 0, alpha = 0.5), "blue", "red"), lty = c(NA, 1, 1),
       pch = c(1, NA, NA), cex = 0.6)
```

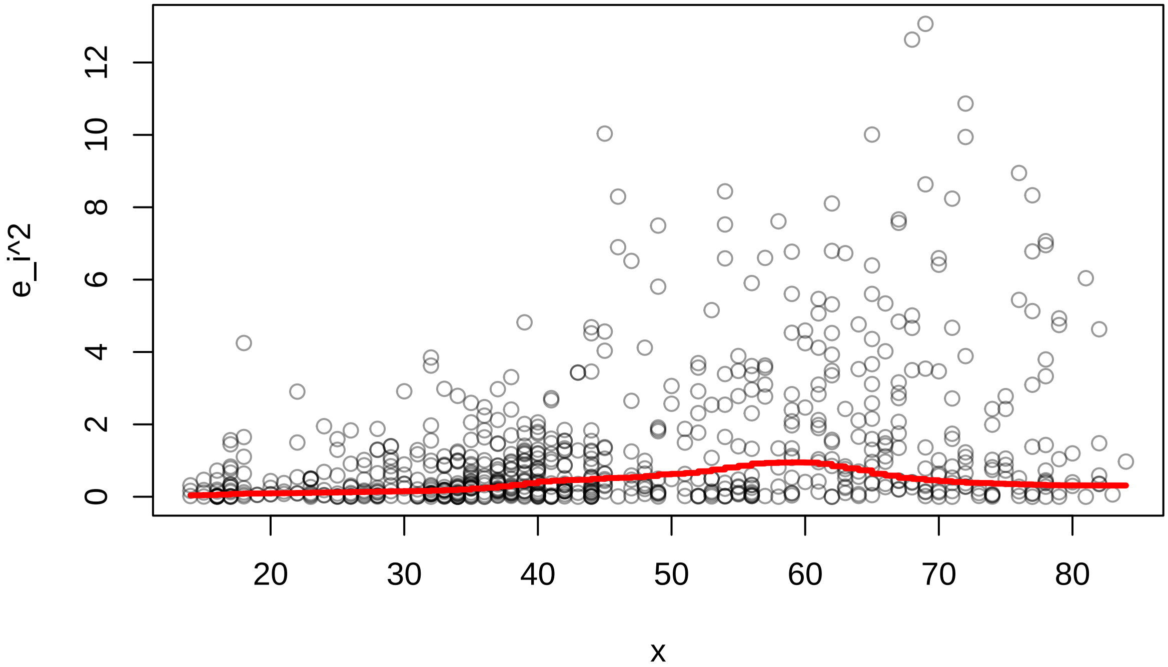


Option 2: using `sm.regression`

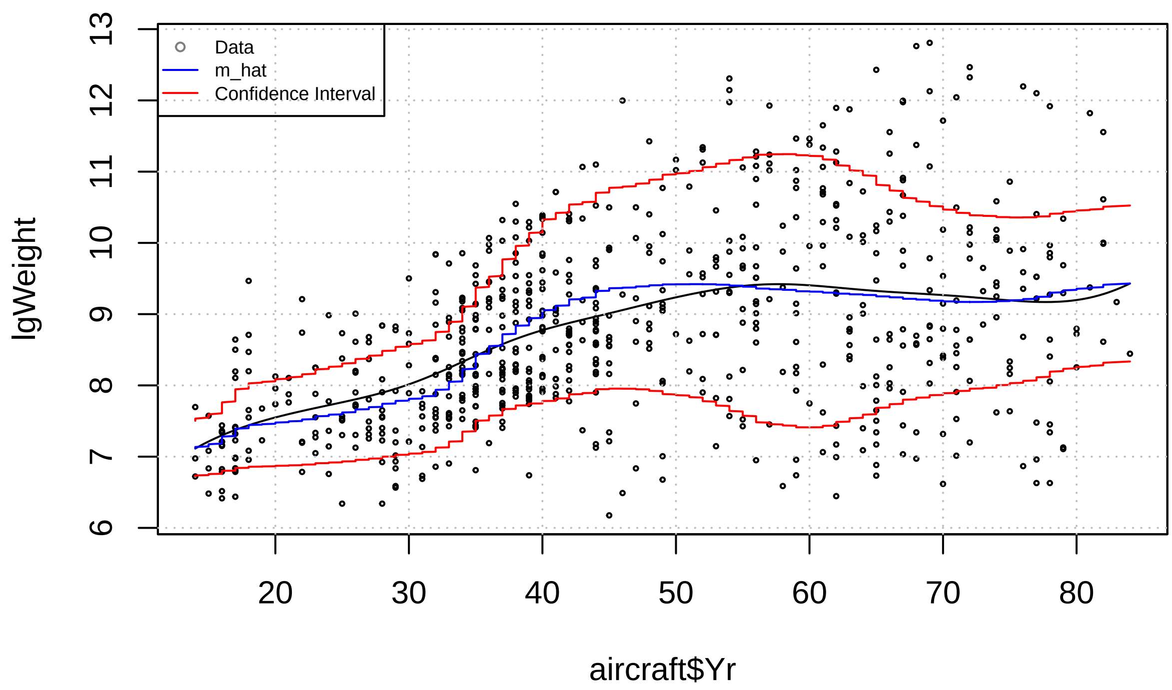
```
# Function sm.regression option
residualss2 <- lgWeight - aircraft.sm_reg$estimate
z_i2 = log(residualss2^2)

h.dpill_zi2 <- dpill(x=aircraft$Yr,y=z_i2,gridsize=length(aircraft$Yr),range.x=range(aircraft$Yr))

q_hat2 = sm.regression(x = aircraft$Yr, y = z_i2, h = h.dpill_zi2, eval.points = seq(min(aircraft$Yr), m
```




```r
sigma_square_hat2 = exp(q_hat2$estimate)


plot(aircraft$Yr, residualss2^2, col = rgb(0, 0, 0, alpha = 0.4), xlab = "x", ylab = "e_i^2")
lines(aircraft$Yr, sigma_square_hat2, col = "red", lwd = 3)  # Adjust line width here
```




```
# Your plot code
set.seed(123)
plot_aux_2 <- sm.regression(x = aircraft$Yr, y = lgWeight, h = h2, eval.points = seq(min(aircraft$Yr), m
# plot(aircraft$Yr, lgWeight, col = rgb(0, 0, 0, alpha = 0.5), xlab = "x", ylab = "m_hat")
lines(aircraft$Yr, aircraft.sm_reg$estimate, type = "l", col = "blue")
lines(aircraft$Yr, aircraft.sm_reg$estimate + 1.96 * sqrt(sigma_square_hat2), col = "red")
lines(aircraft$Yr, aircraft.sm_reg$estimate - 1.96 * sqrt(sigma_square_hat2), col = "red")
grid(col = "gray", lty = "dotted")  # Add grid

# Add legend with adjusted parameters
legend("topleft", legend = c("Data", "m_hat", "Confidence Interval"),
       col = c(rgb(0, 0, 0, alpha = 0.5), "blue", "red"), lty = c(NA, 1, 1),
       pch = c(1, NA, NA), cex = 0.6)
```



1. Conditional Variance

Loading the necessary library and dataset. Storing the logarithm transform of the variables.

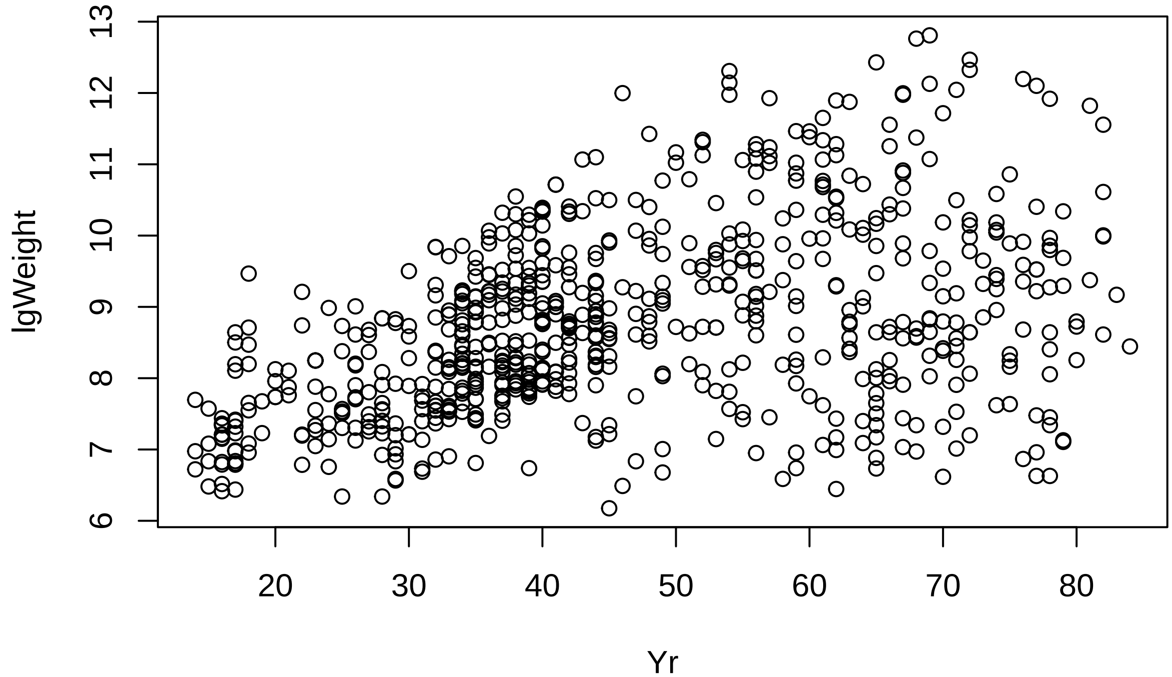
```r
library(sm)
data(aircraft)
help(aircraft)
attach(aircraft)
```

```
## The following objects are masked from aircraft (pos = 4):
##
##     Length, Period, Power, Range, Span, Speed, Weight, Yr
```

```r
lgPower <- log(Power)
lgSpan <- log(Span)
lgLength <- log(Length)
lgWeight <- log(Weight)
lgSpeed <- log(Speed)
lgRange <- log(Range)
```

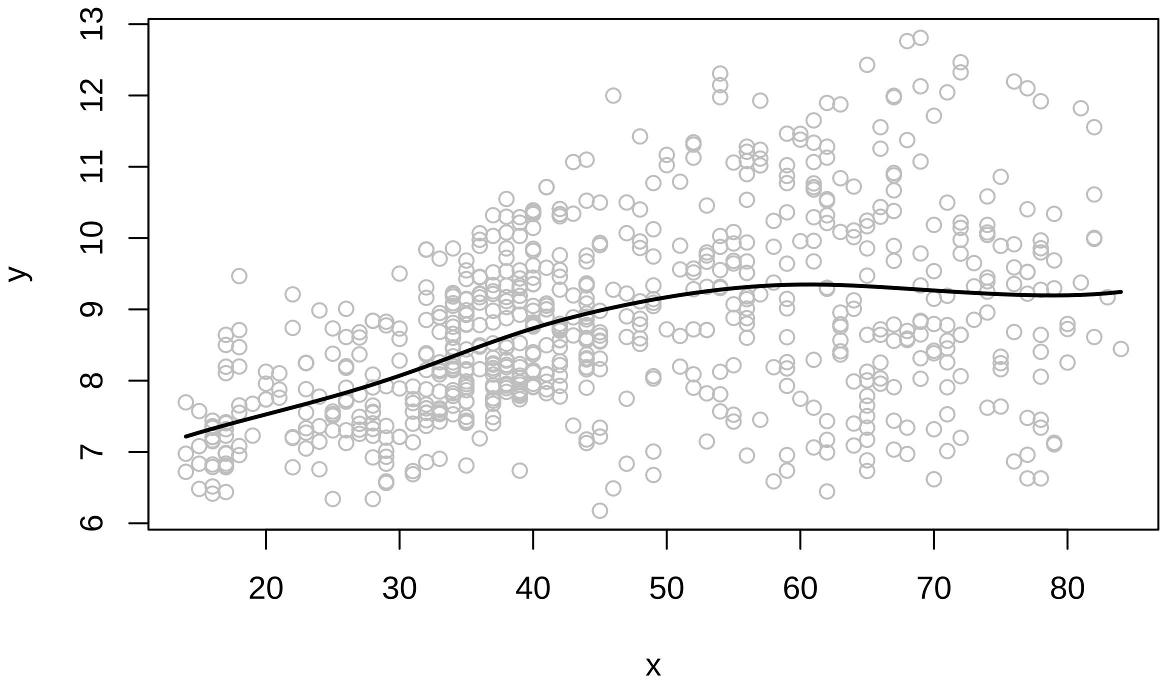
We consider a heteroscedastic regression model $Y = m(X) + \sigma(X)\varepsilon$ for ε being standard, zero-mean Gaussian noise.

```r
plot(Yr, lgWeight)
```

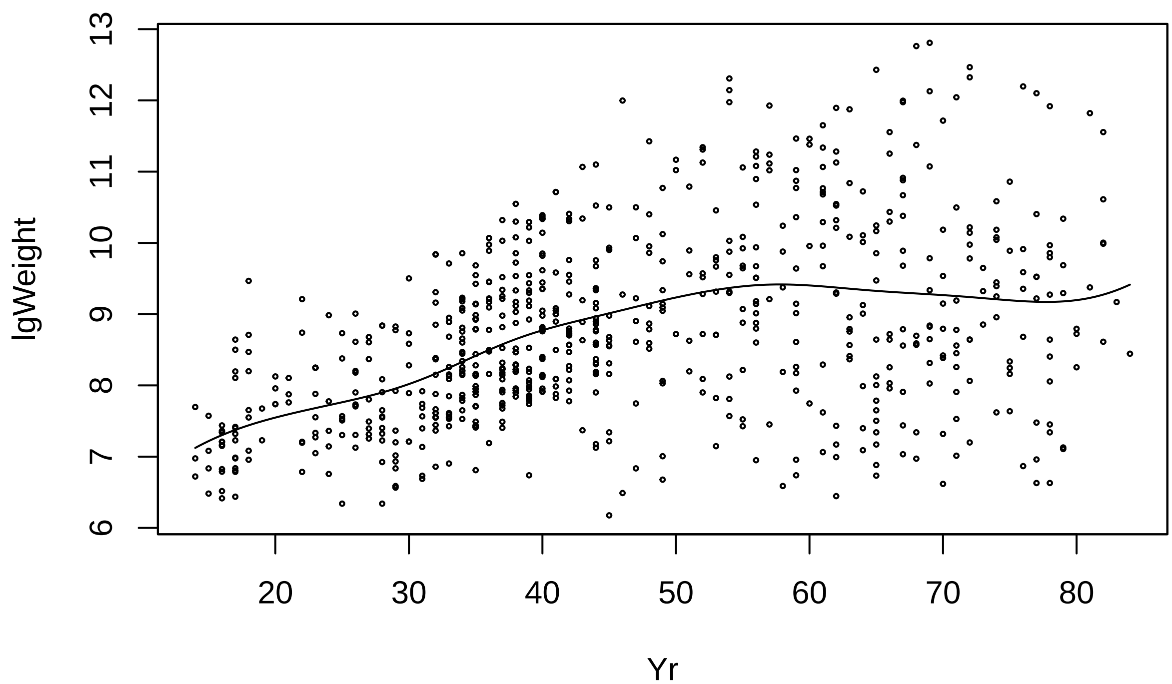


1.1. Nonparametric regression model on the original data

```r
# Function loc.pol.reg option
source("locpolreg.R")
aircraft.lp_reg <- locpolreg(Yr, lgWeight)
```




```r
# Function sm.regression option
library(KernSmooth)
h_m <- dpill(Yr, lgWeight)
aircraft.sm_reg <- sm.regression(Yr, lgWeight, h_m, eval.points = seq(min(Yr), max(Yr), length.out = len
```



1.2. Transformed estimated residuals

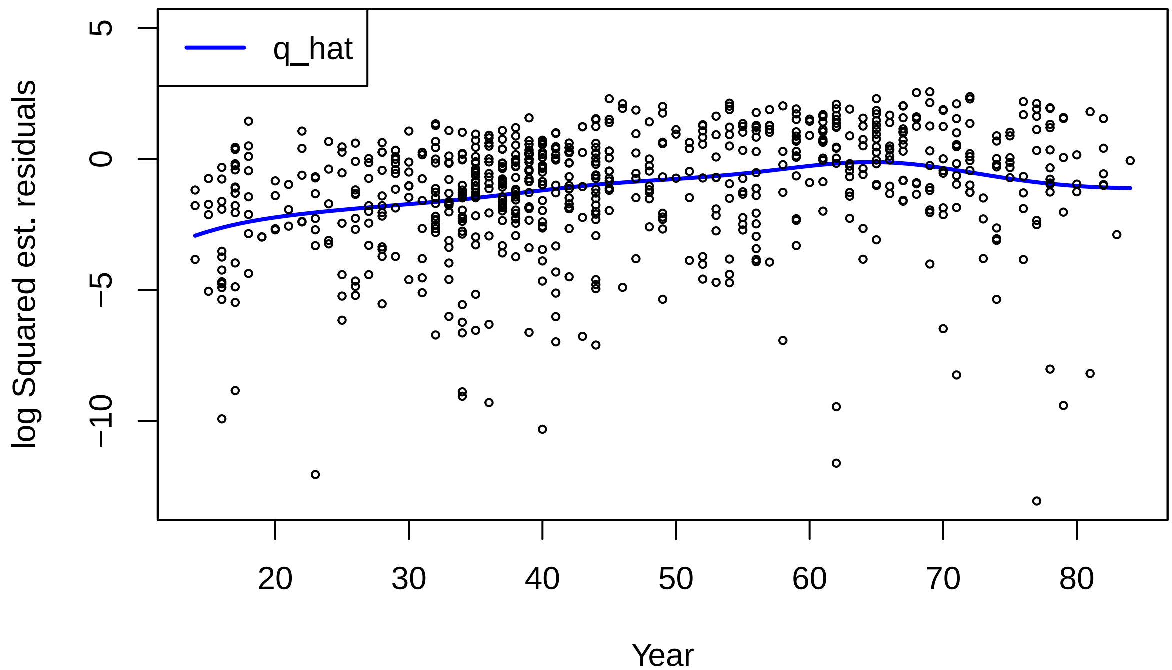
```
yhat.sm <- aircraft.sm_reg$estimate
epsilon.sm <- (lgWeight - yhat.sm)^2
z.sm <- log(epsilon.sm)
```

1.3. Nonparametric regression model for (x_i, z_i)

We'll call the estimated function $\hat{q}(x)$ and save the estimated values in `q_hat`.

The function $\hat{q}(x)$ is an estimate of $\log \sigma^2(x)$.

```
h2_sm <- dpill(Yr, z.sm)
aircraft.sm_reg2 <- sm.regression(Yr, z.sm, h2_sm,
                                 eval.points =seq(min(Yr), max(Yr), length.out=length(Yr)),
                                 ylim = c(min(z.sm), 5),
                                 xlab = "Year", ylab = "log Squared est. residuals",
                                 lwd=2, col="blue", cex=0.5)

legend("topleft", legend = "q_hat", col = "blue", lty = 1, lwd=2)
```




```
q_hat = aircraft.sm_reg2$estimate
```

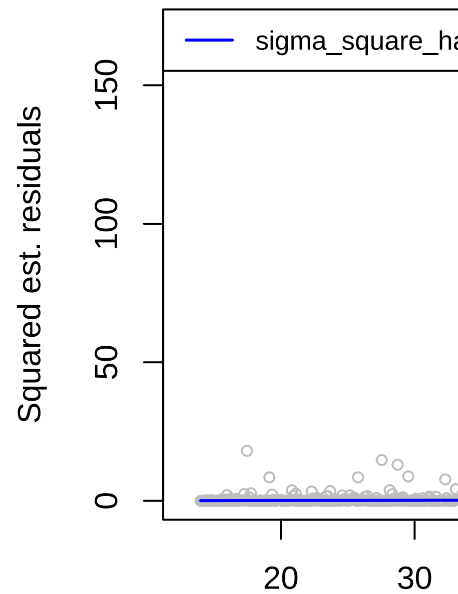
1.4. Estimating $\sigma^2(x)$

We shall estimate the variance by $\hat{\sigma}^2(x) = e^{\hat{q}(x)}$ and save the estimated values in `sigma_square_hat`

```
sigma_square_hat = exp(q_hat)
```

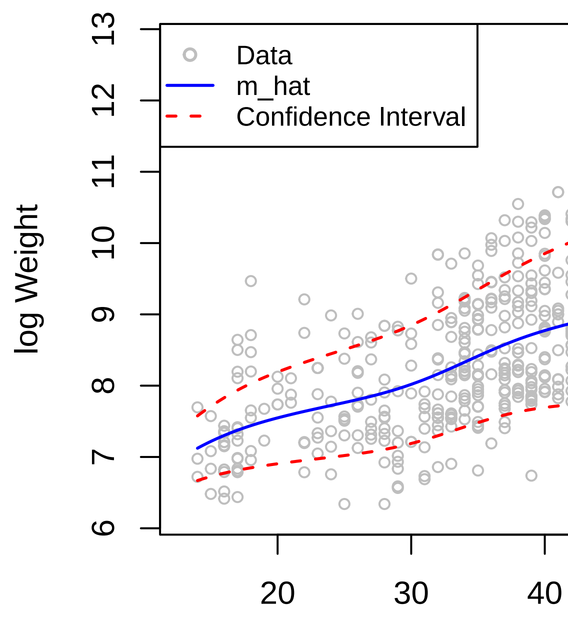
Plots

```
year.points = aircraft.sm_reg$eval.points

# Plot squared residuals against x_i
plot(year.points, epsilon.sm^2, xlab = "Year", ylab = "Squared est. residuals", col="grey", cex=0.7)

# Superimpose the estimated function sigma_square_hat
lines(year.points, sigma_square_hat, col = "blue",lwd=1.5)

legend("topleft", legend = "sigma_square_hat", col = "blue", lty = 1, cex = 0.8,lwd=1.5)
```



Draw a graph of $\hat{\epsilon}_i^2$ against x_i and superimpose the estimated function $\hat{\sigma}^2(s)$.

```
# Plot the estimate m_hat
plot(Yr, lgWeight, xlab = "Year", ylab = "log Weight", col="grey", cex = 0.7)
lines(year.points, yhat.sm, type = "l", col = "blue", lwd=1.5)

# Superimpose the estimated function sigma_square_hat
lines(year.points, yhat.sm + 1.96 * sqrt(sigma_square_hat), col = "red", lty=2, lwd=1.5)
lines(year.points, yhat.sm - 1.96 * sqrt(sigma_square_hat), col = "red", lty=2, lwd=1.5)

# Add legend with adjusted parameters
legend("topleft", legend = c("Data", "m_hat", "Confidence Interval"),
       col = c("grey", "blue", "red"),
       lty = c(NA, 1, 2), pch = c(1, NA, NA), lwd = 1.5, cex = 0.8)
```



Draw the function $\hat{m}(x)$ and superimpose the bands $\hat{m}(x) \pm 1.96\hat{\sigma}(x)$.

2. Local Poisson Regression

Using the dataset from `HDI.2017.subset.csv` for the Human Development Index of nations.

```
data <- read.csv2('HDI.2017.subset.csv')
```

2.1. Bandwidth choice

We modify the functions `loglik.CV` and `h.cv.sm.binomial`to obtain a bandwidth choice method for local Poisson regression based on LOOCV for the expected likelihood. To do so, we will be using the log-likelihood

$$l_{cv}(h) = \frac{1}{n}\sum_{i=1}^{n} log(\hat{Pr}_h^{-i}(Y = y_i|X = x_i)),$$

where $\hat{Pr}_h^{-i}(Y = y_i|X = x_i)$ is an estimate for

$$Pr(Y = y_i|X = x_i) = e^{-\lambda_i}\frac{\lambda_i^{y_i}}{y_i!},$$

where of course

$$\lambda_i = \mathbb{E}\left[Y|X = x_i\right]$$

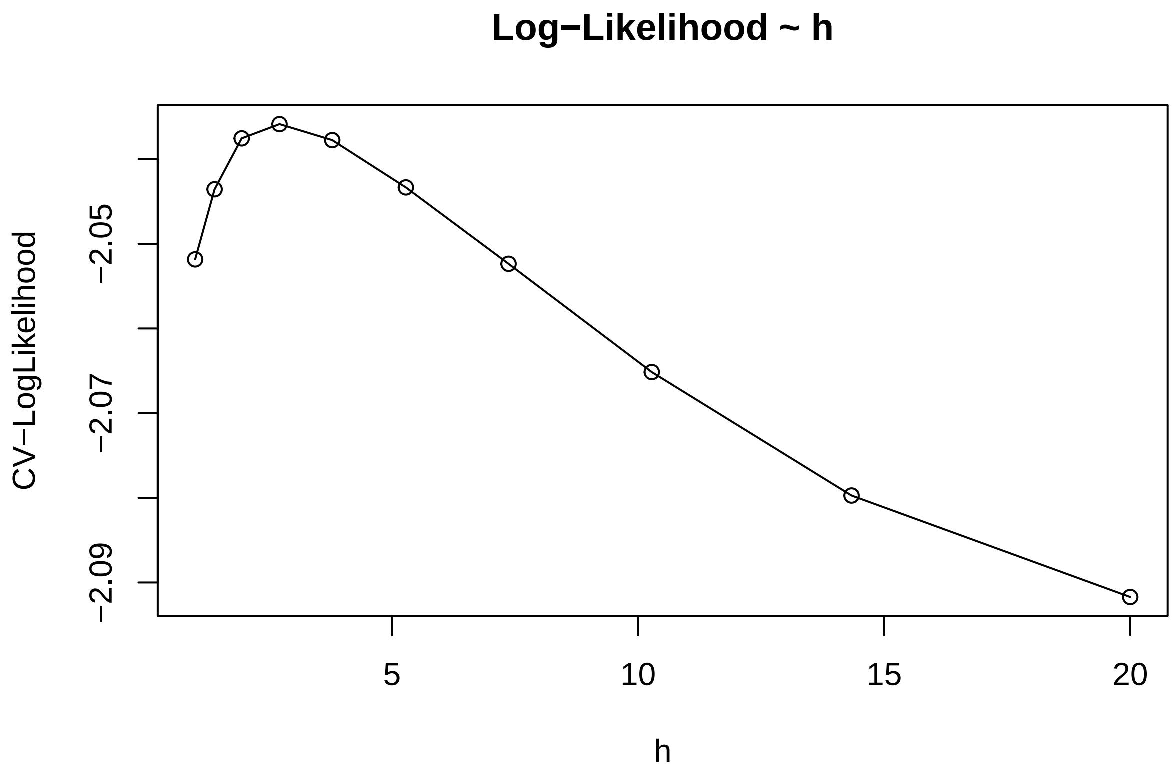
is estimated via maximum likelihood on the hyperparameter `h`.

```r
# Function to estimate the log-likelihood of a Poisson distribution
# via cross-validation
loglik.CV.poisson <- function(X, Y, h){
  n <- length(X)
  pred <- sapply(1:n,
                 function(i, X, Y, h){
                   sm.poisson(x = X[-i], y = Y[-i], h = h, eval.points = X[i],
                              display = "none")$estimate
                 },   X, Y, h)
  like <- exp(-pred)*(pred^Y)/factorial(Y)
  return(mean(log(like)))
}


h.cv.sm.poisson <- function(X, Y, h.range=NULL, l.h=10, method=loglik.CV.poisson){
  cv.h <- numeric(l.h)
  if (is.null(h.range)) {
    hh <- c(h.select(X, Y, method = "cv"),
            h.select(X, Y, method = "aicc"))
    h.range <- range(hh)*c(1/1.1, 1.5)
  }
  i <- 0
  gr.h <- exp(seq(log(h.range[1]), log(h.range[2]), l = l.h))
  for (h in gr.h) {
    i <- i + 1
    cv.h[i] <- method(X, Y, h)
  }
  return(list(h = gr.h,
              cv.h = cv.h,
              h.cv = gr.h[which.max(cv.h)]))
}
```

2.2. Local Poisson regression for Country Development

We now fit a local Poisson regression for `le.fm.r`, a rounded version of the variable `le.fm`, as a function of `Life.expec`.

```r
options(warn = 2)
le.fm.r <- round(data$le.fm)

h.CV.loglik <- h.cv.sm.poisson(X = data$Life.expec, Y = le.fm.r, h.range = c(1,20),
                               method = loglik.CV.poisson)

plot(h.CV.loglik$h,h.CV.loglik$cv.h, main = "Log-Likelihood ~ h",
     xlab = "h", ylab = "CV-LogLikelihood")
lines(h.CV.loglik$h,h.CV.loglik$cv.h)
```


Log−Likelihood ~ h
CV−LogLikelihood
−2.05
−2.07
−2.09
5
10
15
20
h