Local Linear Regression: Estimating Conditional Variance

Azzarito Domenico, Daniel Reverter, Alexis Vendrix (Improved Solution)

11 November, 2025

Estimating the conditional variance by local linear regression

This document follows the procedure outlined in `statement.pdf` to estimate the conditional variance of aircraft weight based on its year of manufacture.

```
# Libraries
library(sm)          # For sm.regression
library(KernSmooth)   # For dpill (Direct Plug-In)
library(ggplot2)     # For plotting CV curves
source("locpolreg.R") # Assuming this file is in the same directory
```

1. Aircraft data and log transformation

We load the `aircraft` dataset and create the required log-transformed variables. We will use $x = \text{Yr}$ and $Y = \text{lgWeight}$ as our variables.

```
# Load data
data(aircraft)

# Define variables for clarity, avoiding attach()
x <- aircraft$Yr
y <- log(aircraft$Weight)

# Define a smooth grid for plotting our final functions
eval.grid <- seq(min(x), max(x), length.out = 200)
```

2. Procedure overview

We follow the 4-step procedure from the documentation to estimate $\sigma^2(x)$ in the model $Y = m(x) + \sigma(x)\epsilon$.

1. Fit $\hat{m}(x)$ to (x_i, y_i).
2. Calculate residuals $\hat{\epsilon}_i = y_i - \hat{m}(x_i)$ and transform them: $z_i = \log(\hat{\epsilon}_i^2)$.
3. Fit $\hat{q}(x)$ to (x_i, z_i).
4. Estimate $\hat{\sigma}^2(x) = e^{\hat{q}(x)}$.

We will perform this procedure twice as requested.

---

3. Part 1: Using `loc.pol.reg` with LOOCV

First, we use the loc.pol.reg function. We will select the optimal bandwidth (the h parameter, which controls smoothness) using Leave-One-Out Cross-Validation (LOOCV). The bandwidth h that gives the best average prediction is chosen.

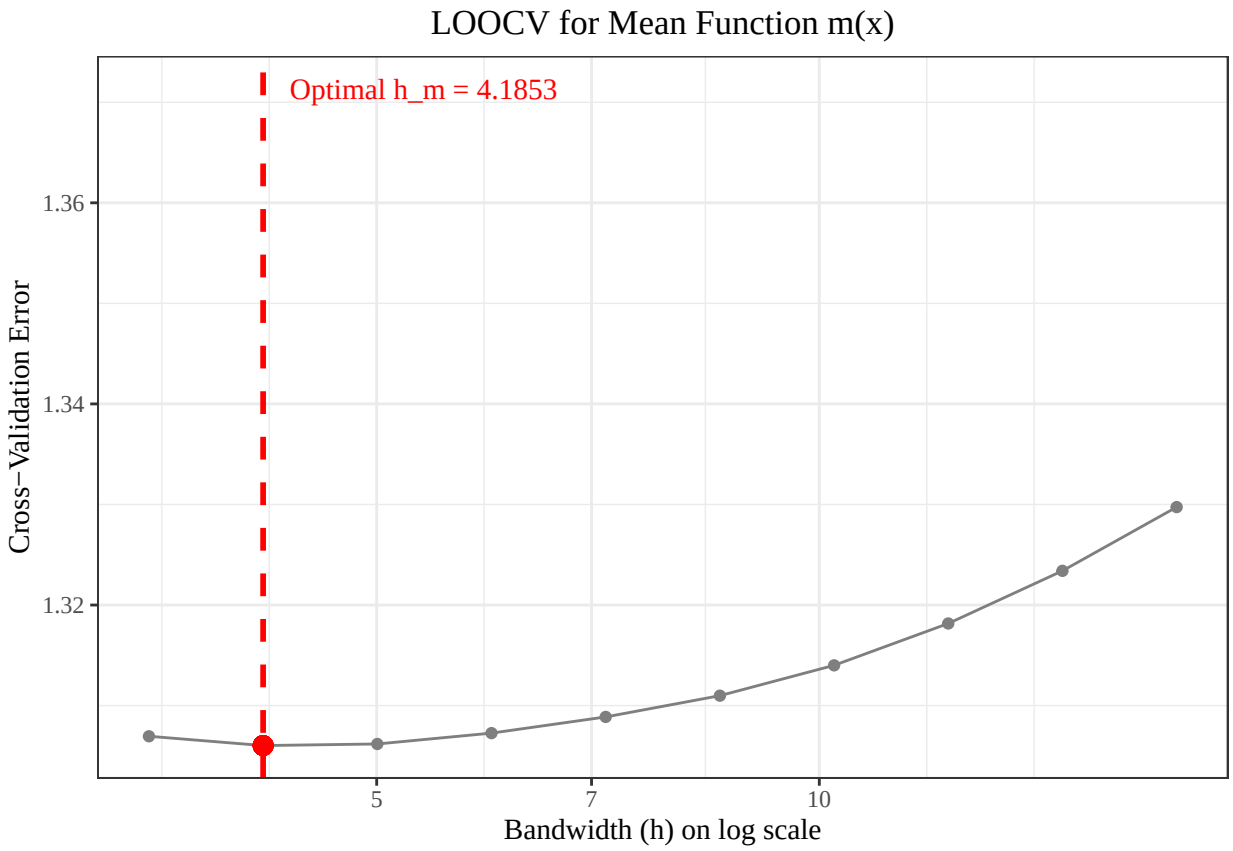
CV Function

Here is the helper function for LOOCV.

```
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$S
    h.y <- aux$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))
}
```

Step 1.1: Fit mean function $\hat{m}(x)$

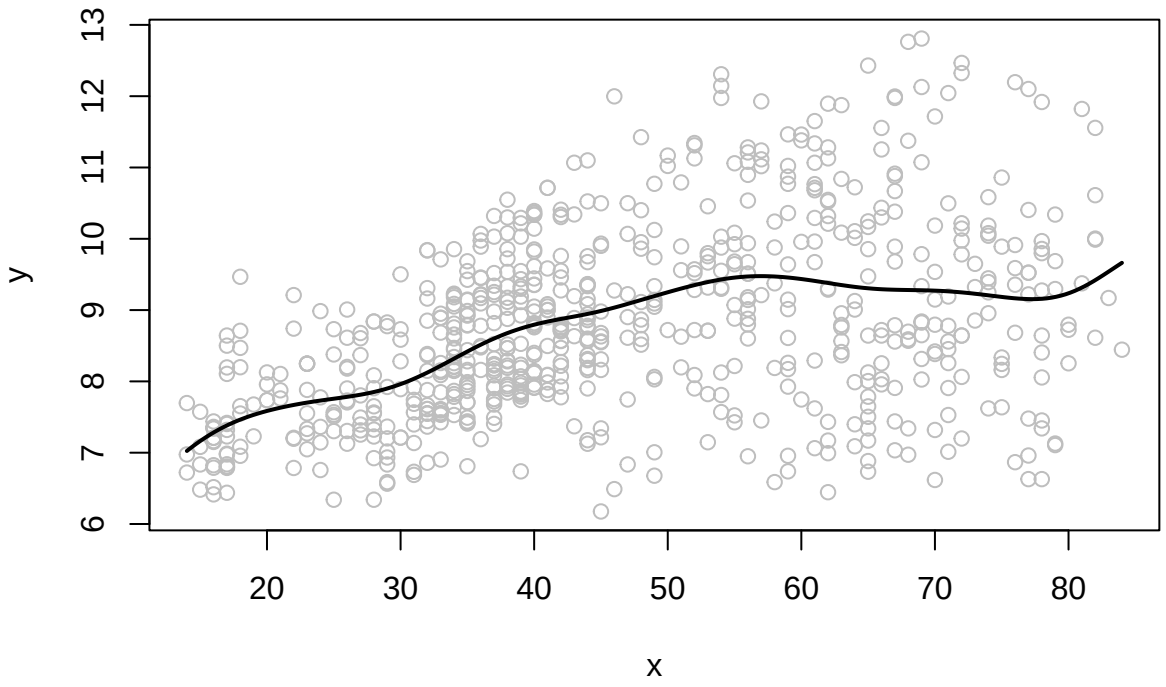
We select the optimal bandwidth h_m for the mean function $m(x)$ using LOOCV. The plot shows the cross-validation error for different bandwidths; we pick the h that minimizes this error.



The optimal bandwidth for the mean function is $h_m = 4.1853$.

Now, we fit $\hat{m}(x)$ on our smooth `eval.grid` for plotting.

```
m.hat.fit <- locpolreg(x, y, h = h.m, tg = eval.grid)
```




```
m.hat.grid <- m.hat.fit$mtgr
```

Step 1.2: Calculate residuals z_i

To get residuals, we must fit $\hat{m}(x_i)$ at the **original data points** x_i, not the grid. This gives us one residual $\hat{\epsilon}_i$ for each data point (x_i, y_i).

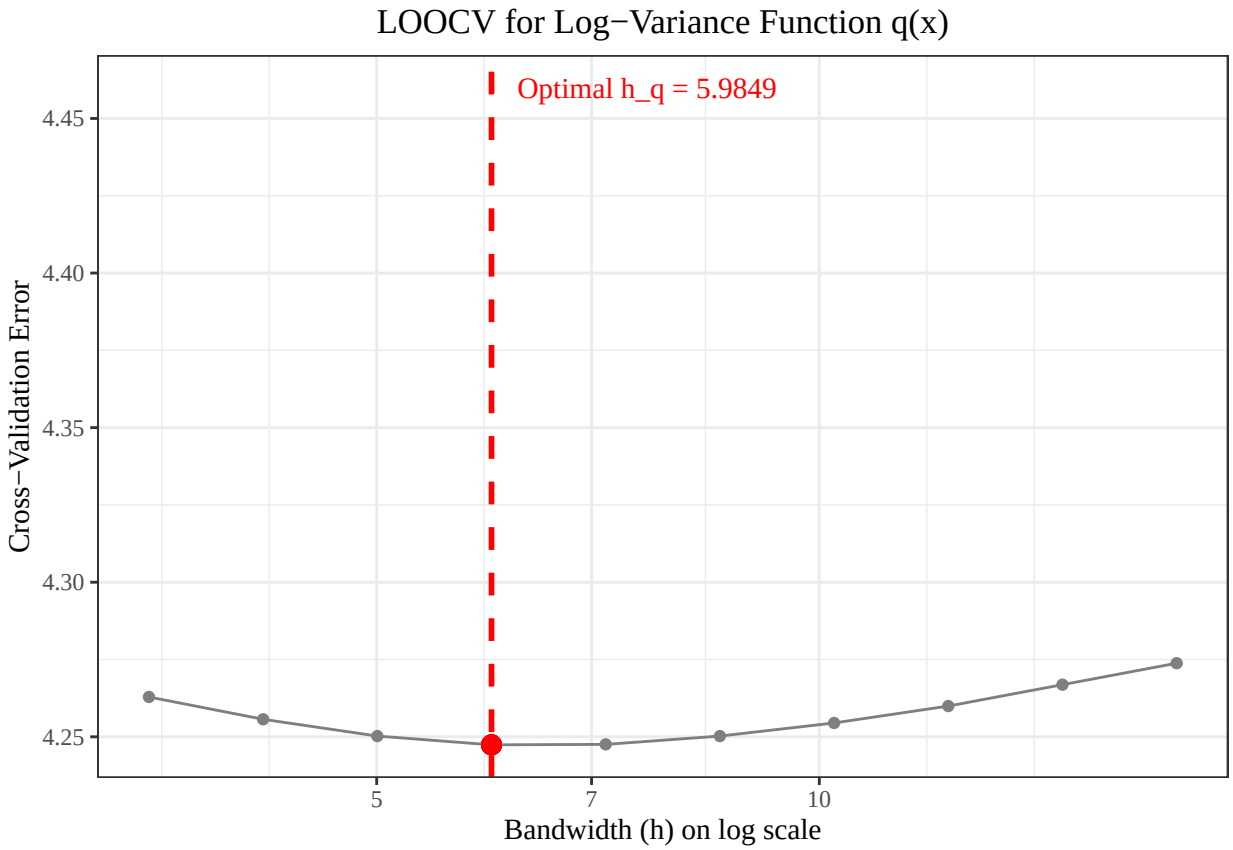
```
m.hat.points <- locpolreg(x, y, h = h.m, tg = x)$mtgr
```

```
eps.hat <- y - m.hat.points
z <- log(eps.hat^2)
```

Step 1.3: Fit log-variance function $\hat{q}(x)$

Now we repeat the bandwidth selection, but this time for the new regression of $z_i = \log(\hat{\epsilon}_i^2)$ on x_i. This will give us a new optimal bandwidth, h_q.

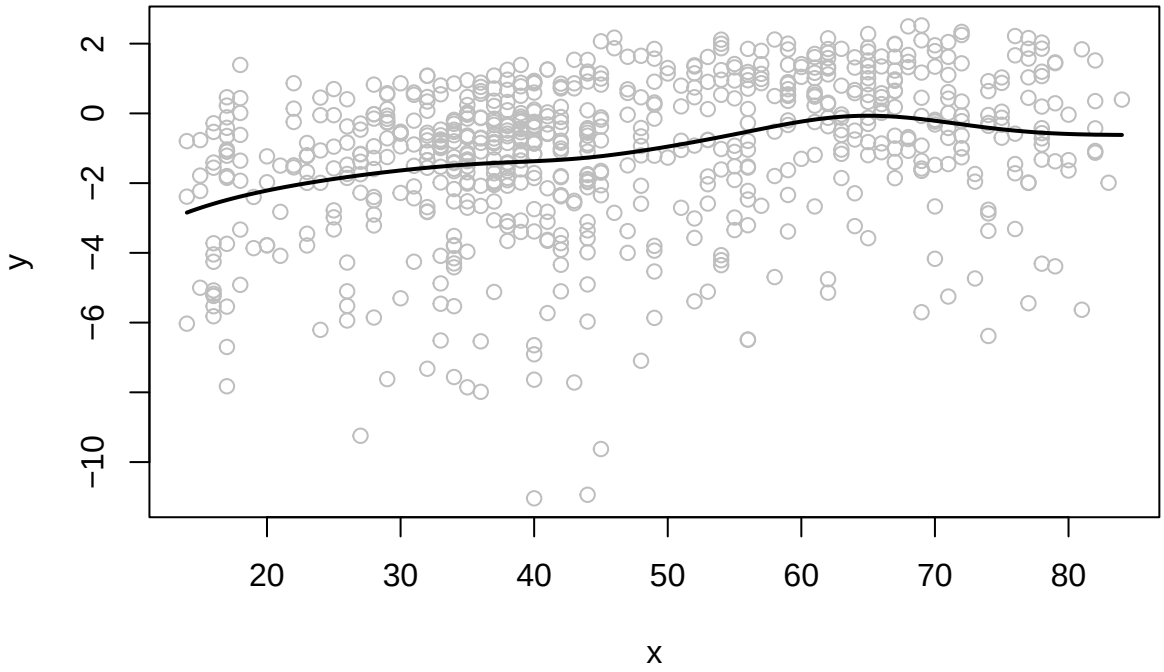
We fit a nonparametric regression to (x_i, z_i) to get $\hat{q}(x)$. We find a new optimal bandwidth h_q using LOOCV.



The optimal bandwidth for the log-variance function is $h_q = 5.9849$.

Now, we fit $\hat{q}(x)$ on our smooth `eval.grid` for plotting.

```
# Fit q(x) on the smooth grid for plotting
q.hat.fit <- locpolreg(x, z, h = h.q, tg = eval.grid)
```



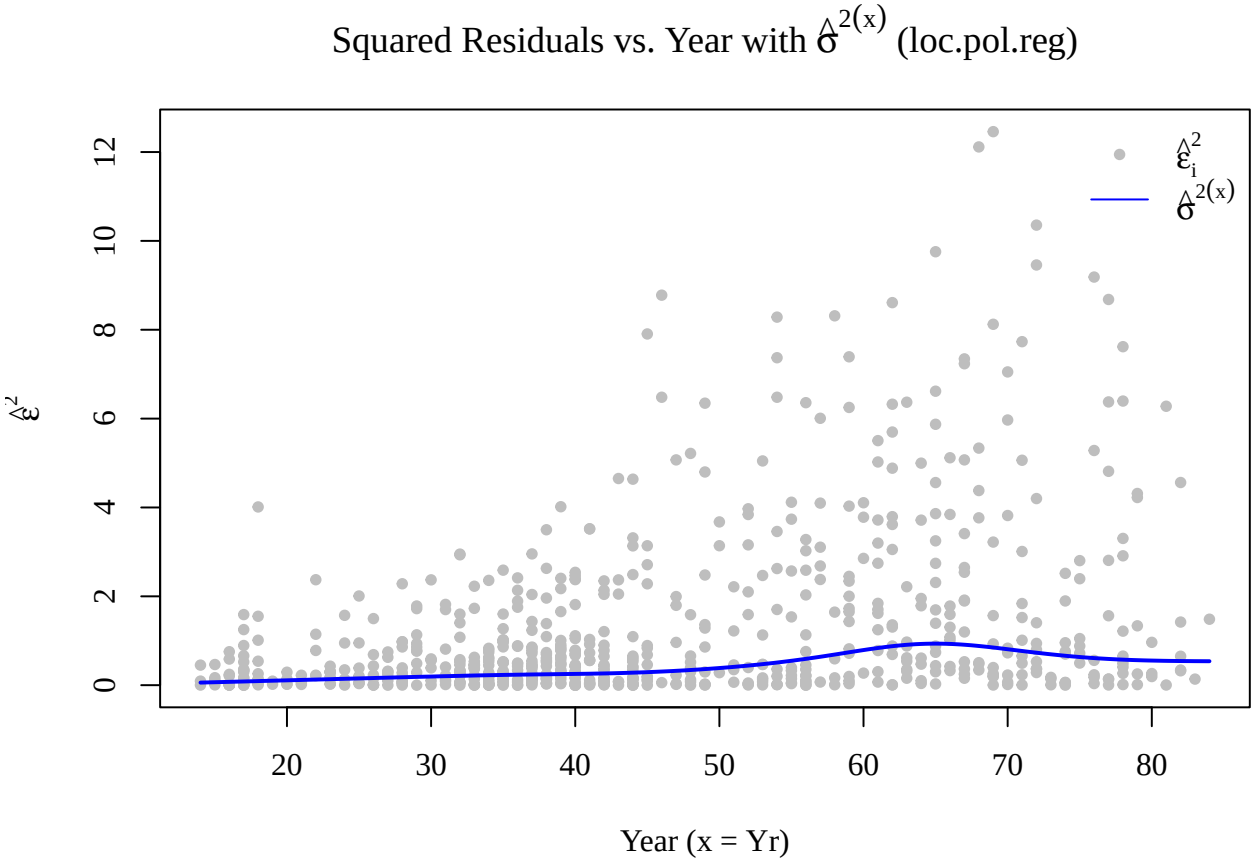
```
q.hat.grid <- q.hat.fit$mtgr
```

Step 1.4: Estimate $\hat{\sigma}^2(x)$ and plot

We complete the procedure by exponentiating $\hat{q}(x)$ to get our final variance estimate $\hat{\sigma}^2(x)$ and our standard deviation estimate $\hat{\sigma}(x)$.

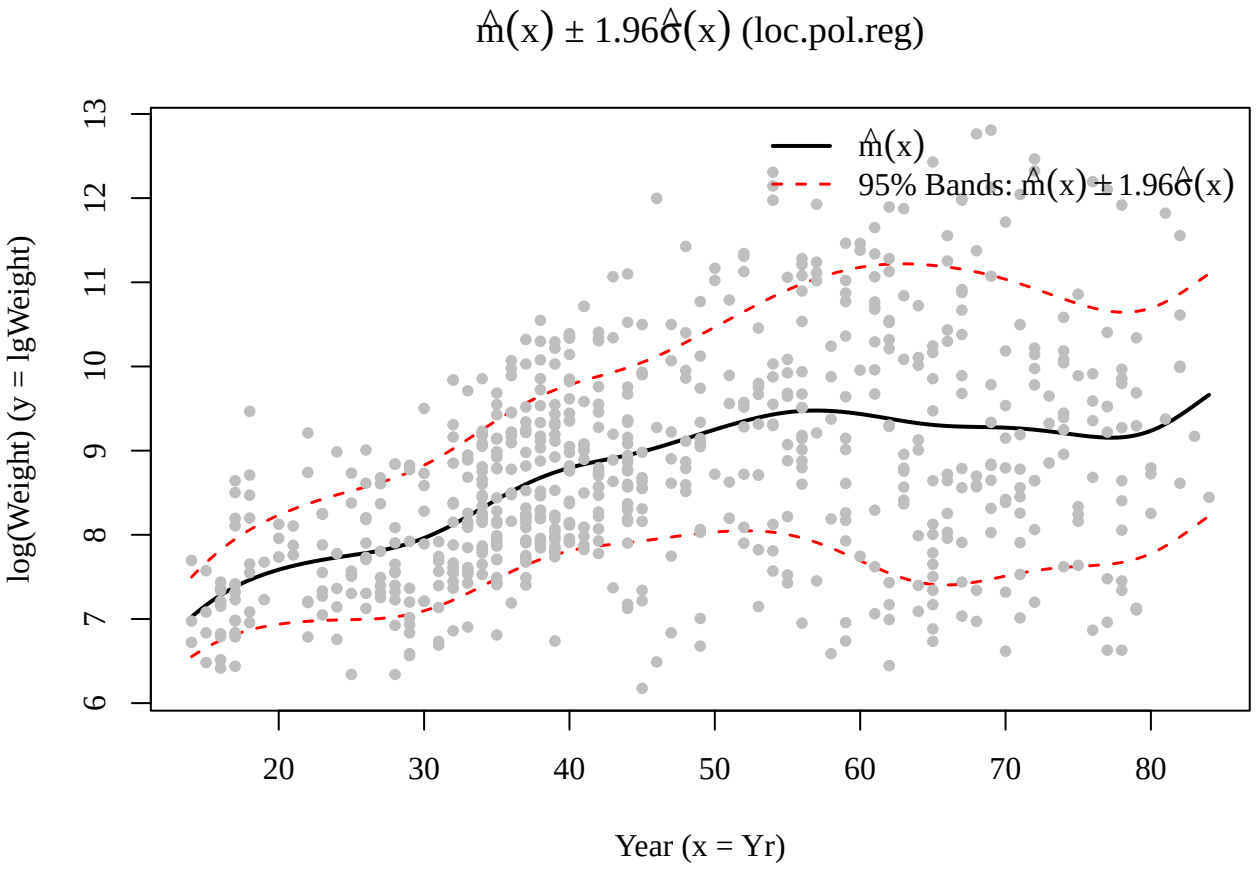
```
# Get variance and standard deviation estimates on the grid
sigma2.hat.grid <- exp(q.hat.grid)
sigma.hat.grid  <- sqrt(sigma2.hat.grid)
```

Plot 1: Squared residuals vs. Year



This plot shows the "noisy" variance proxies (the squared residuals $\hat{\epsilon}_i^2$ in grey) against the year. The solid blue line is our smooth estimate of the underlying conditional variance function, $\hat{\sigma}^2(x)$. This line captures the trend in the variance.

Plot 2: Mean function $\hat{m}(x)$ with 95% bands



4. Part 2: Using `sm.regression` with DPI

Second, we repeat the entire procedure using the `sm.regression` function from the `sm` library. The key difference here is the bandwidth selection. Instead of the slow LOOCV, we use `dpill` from the `KernSmooth` library. Direct Plug-In (DPI) is an automatic method that estimates the optimal bandwidth based on the data's properties (like its curvature) rather than iterating through many possibilities.

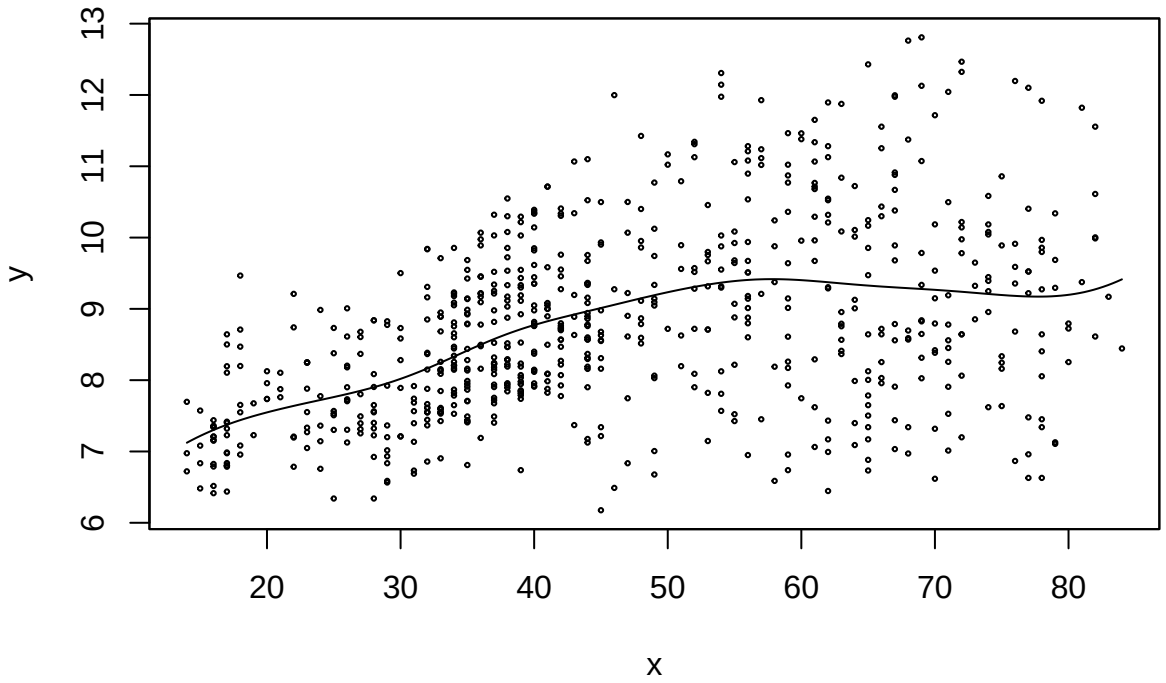
Step 2.1: Fit mean function $\hat{m}(x)$

We select h_m using `dpill` and fit $\hat{m}(x)$ on the smooth `eval.grid`.

```
h.m.sm <- dpill(x, y)
cat(sprintf("DPI bandwidth for m(x) is h_m = %.4f\n", h.m.sm))
```

DPI bandwidth for m(x) is h_m = 5.6509

```
# Fit m(x) on the smooth grid for plotting
sm.m.fit <- sm.regression(x, y, h = h.m.sm, eval.points = eval.grid, model = "none")
```

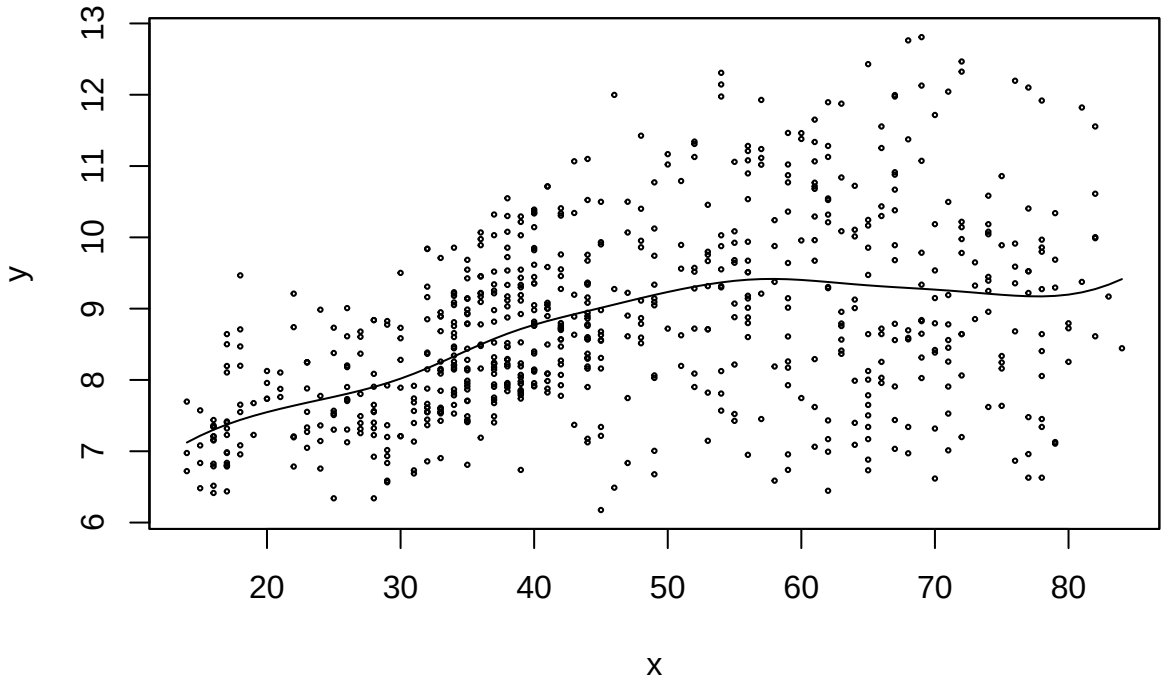



```
m.hat.sm.grid <- sm.m.fit$estimate
```

Step 2.2: Calculate residuals z_i

To get residuals, we must evaluate the fit at the **original data points** x_i.

```
# Fit m(x) at the original data points (eval.points=x) to get residuals
sm.m.points <- sm.regression(x, y, h = h.m.sm, eval.points = x, model = "none")
```




```
eps.hat.sm <- y - sm.m.points$estimate
z.sm <- log(eps.hat.sm^2) # Transformed residuals
```

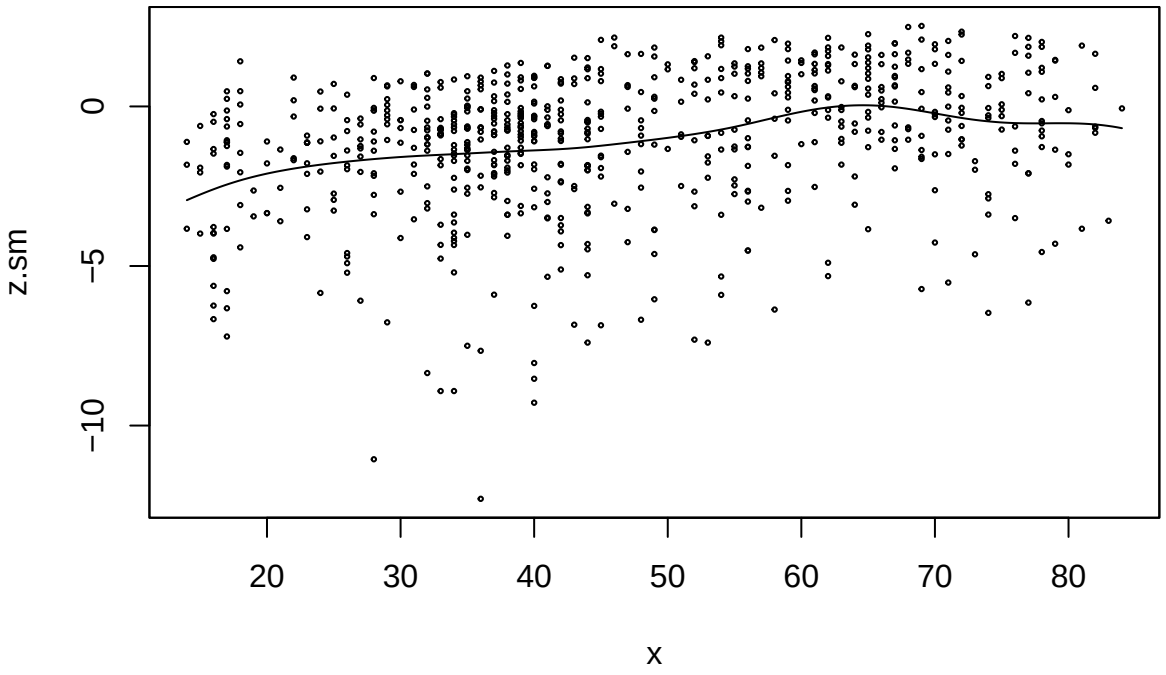
Step 2.3: Fit log-variance function $\hat{q}(x)$

We select h_q using `dpill` for the (x_i, z_i) data and fit $\hat{q}(x)$ on the smooth `eval.grid`.

```
h.q.sm <- dpill(x, z.sm)
cat(sprintf("DPI bandwidth for q(x) is h_q = %.4f\n", h.q.sm))
```

DPI bandwidth for q(x) is h_q = 4.4406

```
# Fit q(x) on the smooth grid for plotting
sm.q.fit <- sm.regression(x, z.sm, h = h.q.sm, eval.points = eval.grid, model = "none")
```



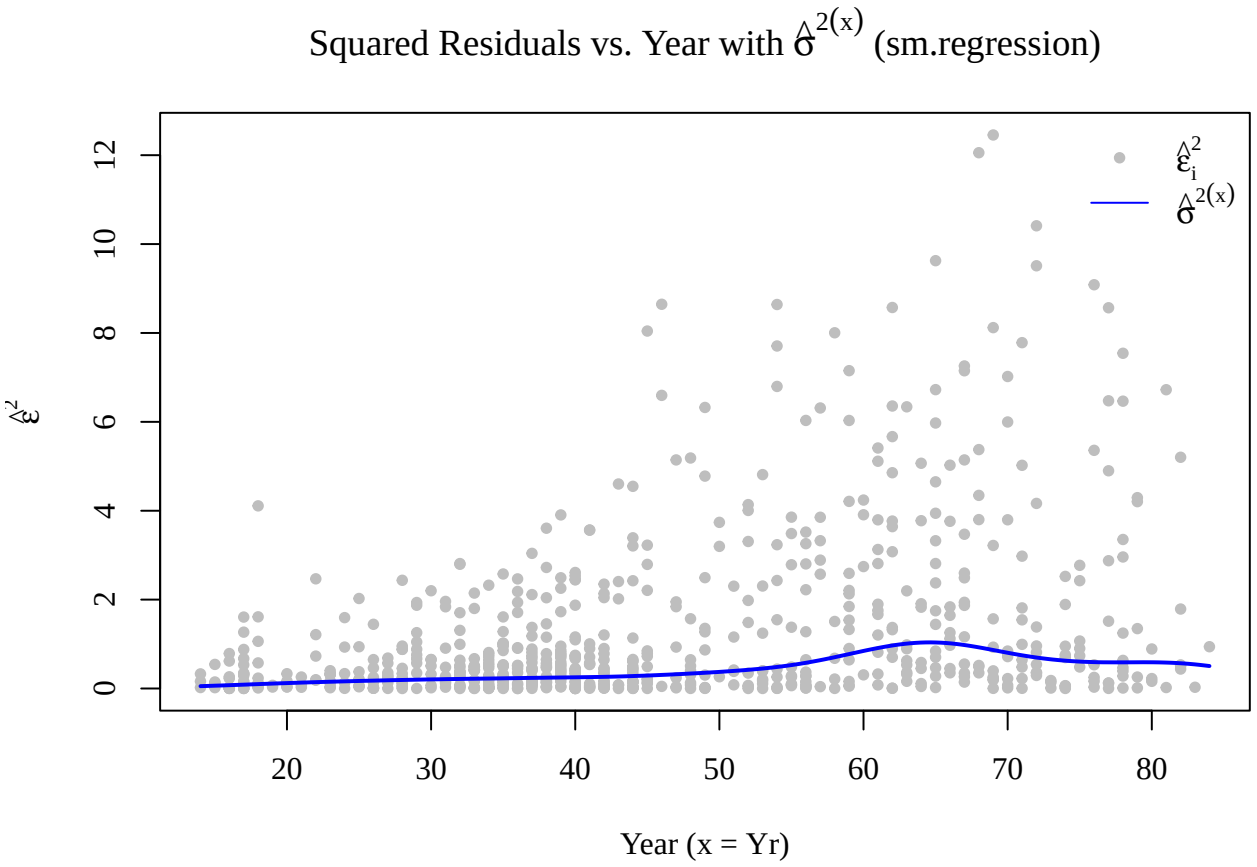

```
q.hat.sm.grid <- sm.q.fit$estimate
```

Step 2.4: Estimate $\hat{\sigma}^2(x)$ and Plot

We estimate $\hat{\sigma}^2(x) = e^{\hat{q}(x)}$ and create the required plots.

```
# Get variance and standard deviation estimates on the grid
sigma2.hat.sm.grid <- exp(q.hat.sm.grid)
sigma.hat.sm.grid  <- sqrt(sigma2.hat.sm.grid)
```

Plot 1: Squared Residuals vs. Year This plot is very similar to the one from Part 1. The grey dots are identical, but the blue line ($\hat{\sigma}^2(x)$) is slightly different due to the different function and bandwidth selector.

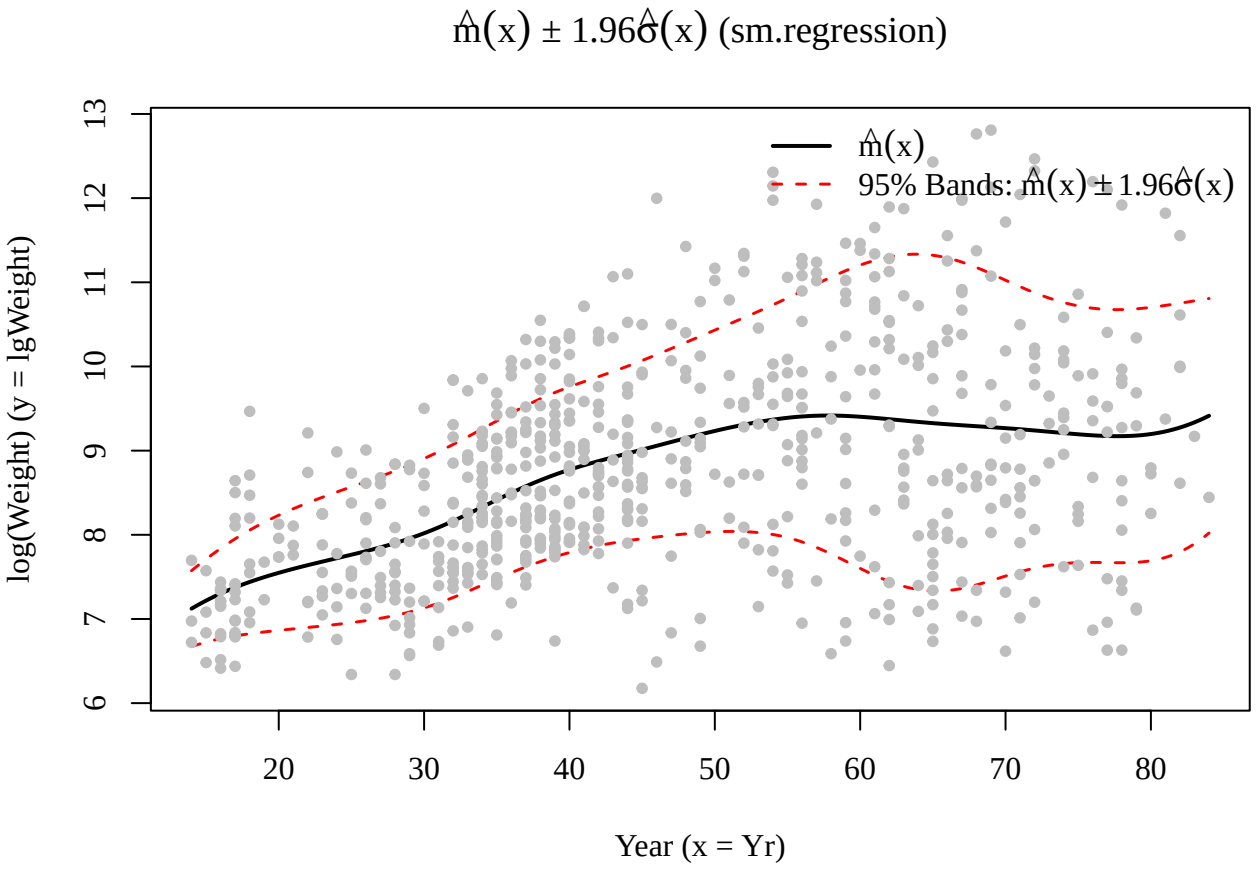


Plot 2: Mean Function $\hat{m}(x)$ with 95% Bands

```
# Calculate bands
upper.band.sm <- m.hat.sm.grid + 1.96 * sigma.hat.sm.grid
lower.band.sm <- m.hat.sm.grid - 1.96 * sigma.hat.sm.grid

# Calculate dynamic y-axis limits
plot.ylim.sm <- range(y, upper.band.sm, lower.band.sm)

par(family = "serif")
# Plot the smooth line using the grid estimates
plot(eval.grid, m.hat.sm.grid, type="l", lwd=2, col="black",
     main=expression(paste(hat(m)(x), " \U00B1 1.96", hat(sigma)(x), " (sm.regression)")),
     xlab="Year (x = Yr)", ylab="log(Weight) (y = lgWeight)",
     ylim = plot.ylim.sm)
lines(eval.grid, upper.band.sm, col="red", lty=2, lwd=1.5)
lines(eval.grid, lower.band.sm, col="red", lty=2, lwd=1.5)
points(x, y, col="grey", pch=20) # Add raw data
legend("topright",
       legend = c(expression(hat(m)(x)), expression(paste("95% Bands: ",
                                          hat(m)(x) %+-% 1.96*hat(sigma)(x)))),
       lty = c(1,2), col = c("black","red"), lwd = c(2,1.5), bty = "n")
```



5. Comparison of methods: `loc.pol.reg` vs. `sm.regression`

Both procedures were successful in modeling the heteroscedasticity of the data. The final plots in Part 1 and Part 2 are conceptually identical: they both show that the average `lgWeight` increases with `Yr`, and more importantly, that the *variance* also increases, which is visualized by the fanning-out prediction bands.

However, the two methods differ significantly in their approach and computational cost, primarily due to **bandwidth selection**.

Bandwidth selection (LOOCV vs. DPI)

This is the most important distinction between the two parts.

- **Part 1: `loc.pol.reg` with LOOCV**
  - We used Leave-One-Out Cross-Validation (LOOCV), a "brute-force" but intuitive method.
  - **Pro:** It directly tests which bandwidth `h` gives the best out-of-sample prediction error for the specific data we have.
  - **Con:** It could be slow as it requires re-fitting the model many times for each potential bandwidth.
- **Part 2: `sm.regression` with DPI**
  - We used Direct Plug-In (DPI) via the `dpill` function.
  - **Pro:** It is an automatic, formula-based method that estimates the optimal bandwidth from properties of the data (like its estimated curvature). It is **extremely fast**.
  - **Con:** It relies on asymptotic formulas and assumptions, which may not always be a perfect fit for smaller or unusual datasets.

Comparison of selected sandwidths

Because the selection methods are different, they will almost always result in different optimal bandwidths.

Table 1: Comparison of Optimal Bandwidths

Function	h (LOOCV)	h (DPI)
Mean m(x)	4.1853	5.6509
Log-Variance q(x)	5.9849	4.4406