ASM - Local polynomial regression

Sergi Carol, Laura Cebollero, Alex Rodriguez

21st November, 2018

Introduction

In this deliverable we will perform some calculation in order to estimate the conditional variance on a local polynomial regression model. In this case we will use data from different aircrafts over the years, mainly focusing on how does the *weight* of the aircrafts change over the years.

Due to the difference in scale of the raw weight data we will begin by transforming the **weight** attribute into logarithmic scale.

```
lgWeight <- log(Weight)
```

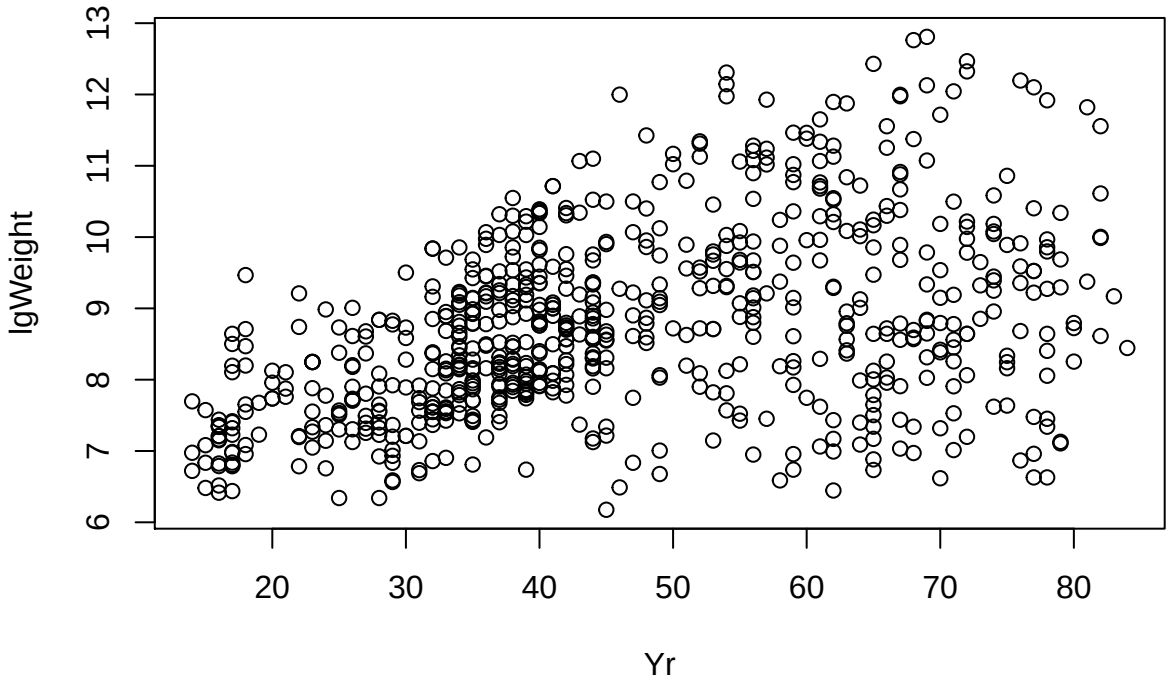
To compute the best bandwidth we have created the function below which computes the mean of the 4 methods we are using to compute an adequate bandwidth.

```
source("functions.R")
# We use the functions to calculate the optimal value for the bandwidth.
# The methods used are cross-validation, generalized cross-validation, 10-fold
# cross-validation and Plug-In
choose.best.bw <- function(x, y){
  cvh.bw <- h.cv.gcv(x, y)
  dpill.bw <- dpill(x, y, range.x = range(x))
  k.cv.bw <- h.k.fold.cv(x, y)

  cv.mean <- mean(cvh.bw[["cv"]])
  gcv.mean <- mean(cvh.bw[["gcv"]])
  kf.cv.mean <- mean(k.cv.bw[["k.cv"]])

  mean.bw = mean(c(cv.mean, gcv.mean, kf.cv.mean, dpill.bw))

  return(mean.bw)
}
```

1. Fitting non-parametric regression data (x, y)

We will begin by simply plotting the Yr against the weight with logarithmic scale.

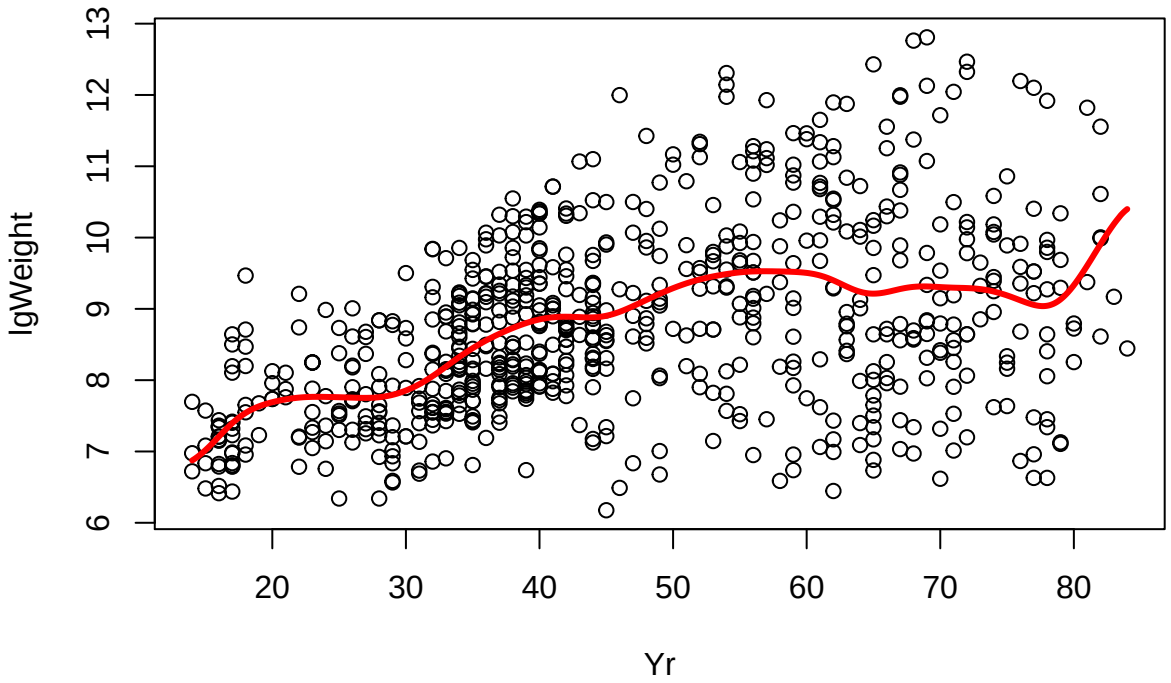
```
plot(Yr, lgWeight)
```



We can see points somewhat follow a curved line, specially on the bottom left of the plot. On the right part of it the plots are more dispersed so visually we are not able to imagine accurately what logistic polynomic regression could fit this data.

Let's compute the regression line using the locpoly method and choosing the best bandwidth with the previously mentioned function we have implemented.

```
plot(Yr, lgWeight)

bw <- choose.best.bw(Yr, lgWeight)
m1 <- locpoly(Yr, lgWeight, bandwidth = bw, degree=1, drv=0, gridsize = length(Yr))
points(m1, col="red", type='l', lwd=3)
```



We can see how the line above could be, at least visually, an adequate representation on what trend the data may follow.

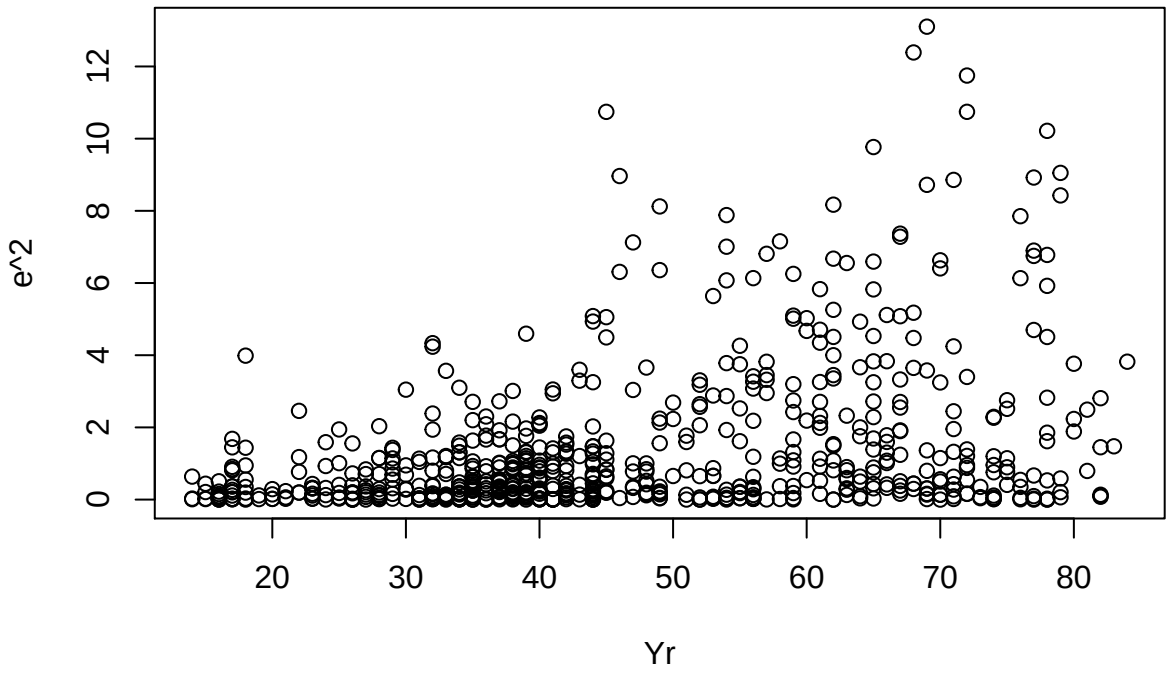
2. Estimated residuals & transformation

Now let's proceed onto computing the Estimated residuals of such polynomic regression:

```
e <- lgWeight - m1$y
summary(e)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.27760 -0.74545 -0.11071 -0.03421  0.65751  3.61971
```

```
# Error compared to our data
plot(Yr, e^2)
```



We can see how most of the residuals are close to 0 along all the years and only a few have higher values, between 4 and 12.

Let's transform the estimated residuals:

```
z <- log(e^2)
summary(z)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -11.0969  -2.0127  -0.6545  -1.0534   0.4800   2.5728
```

```
z[is.infinite(z)] <- 0 # Last value is infinite so we better change it to 0
```

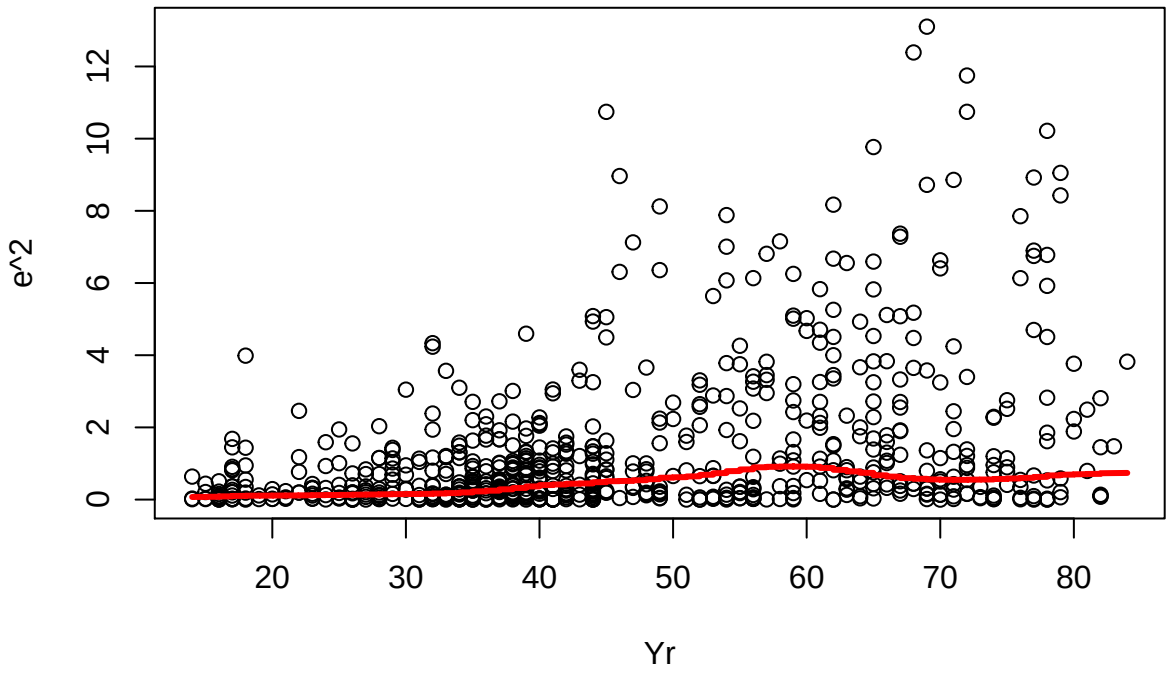
3. Fitting non-parametric regression data (x, z)

Now, we are going to fit the non parametric regression data with the new computed transformed residual.

```
# Choose Bandwidth
bw <- choose.best.bw(Yr, z)
q <- locpoly(Yr, z,  bandwidth = bw, degree=1, drv=0, gridsize = length(Yr))
est.y <- exp(q$y)
summary(est.y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07093 0.18585 0.43597 0.44313 0.63539 0.91860
```

```
# How close is our estimator
plot(Yr, e^2)
points(Yr, est.y, col="red", type='l', lwd=3)
```



We can see how the estimatd polynomic regression fits the transformed residuals.

4. Standard deviation estimation

Finally, let's estimate the standard deviation.

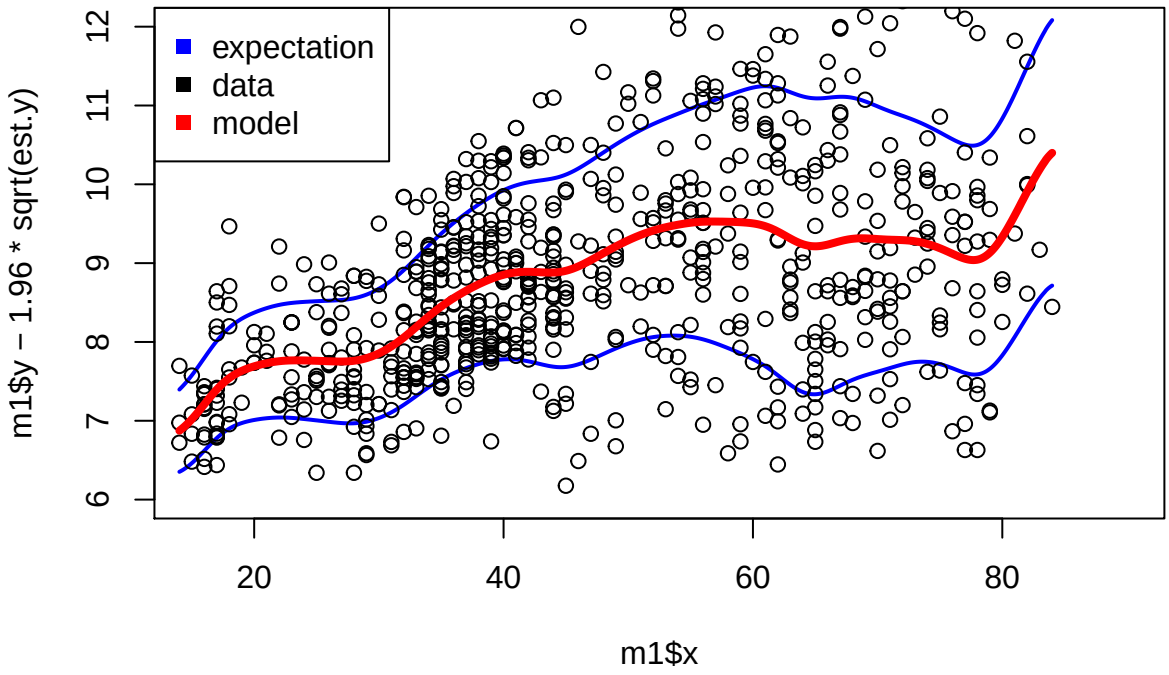
```
est.x <- exp(q$x)
summary(est.x)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 1.203e+06 4.789e+13 1.907e+21 4.532e+34 7.596e+28 3.025e+36
```

We can see how it varies a lot on the weight scale.

Let's see how this standard deviation plots against the real data and with which margin our predictions are fit and may vary:

```
# How differnet is our error
plot(m1$x, m1$y - 1.96*sqrt(est.y), col="blue", type='l', lwd=2, xlim=c(15,90), ylim=c(6,12)) # Check t
points(m1$x, m1$y + 1.96*sqrt(est.y), col="blue", type='l', lwd=2) # Check this bc wtf is this plot may

points(Yr, lgWeight)
points(m1, col="red", type='l', lwd=4)
legend("topleft", c("expectation", "data", "model"), col=c("blue", "black", "red"), pch=15)
```



We can see as we commented on the first section how the standard deviation is greater on the right side of the plot because the points are more scattered, whereas on the left side of the plot they are near the same area.