ASM - Local Poisson regression

Sergi Carol, Laura Cebollero, Alex Rodriguez

10th November, 2018

1 Introduction

In this delivery we are asked to study local Poisson Regression. More specifically, the aim of this report is two-fold:

- Create a function to choose the best bandwidth for a Poisson.
- Use said function on the Country Development dataset.

2 Bandwidth choice for local Poisson Regression

We have modified the already seen functions so that they can be used with data following a Poisson distribution.

To do so, we have had to change the leave-one-out CV estimation of the expected log-likelihood following the formula given in the delivery statement. The resulting code of such function is:

```
loglik.CV <- 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)
    total_sum = 0
    for (i in 1:n){
      total_sum = total_sum + (log((exp(pred[i]) * ((pred[i]**y[i])/factorial(y[i])))))/n
    }
    return(total_sum)
}
```

Now, since we have the log-likelihood method for the Poisson distribution, we can compute the best bandwidth using it:

```
h.cv.sm.poisson <- function(x,y,rg.h=NULL,l.h=10,method=loglik.CV){
  cv.h <- numeric(l.h)
  if (is.null(rg.h)){
    hh <- c(h.select(x,y,method="cv"),
            h.select(x,y,method="aicc"))#,hcv(x,y))
    rg.h <- range(hh)*c(1/1.1, 1.5)
  }
  i <- 0
  gr.h <- exp( seq(log(rg.h[1]), log(rg.h[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,
```

```r
              h.cv = gr.h[which.min(cv.h)]))
}
```

Now that we have them implemented we can proceed to use them for data following a Poisson.

3 Local Poisson regression for Country Development data

Let's first load the countries data:

```r
countries <- na.omit(read_table2("countries.csv"))
```

```
## Parsed with column specification:
## cols(
##   Row = col_integer(),
##   Country = col_character(),
##   life.exp = col_integer(),
##   agricul = col_character(),
##   inf.mort = col_integer(),
##   life.exp.f = col_integer(),
##   life.exp.m = col_integer(),
##   le.fm = col_integer()
## )
```

```r
summary(countries)
```

```
##       Row           Country             life.exp      agricul
##  Min.   :  1.00   Length:132         Min.   :39.0   Length:132
##  1st Qu.: 33.75   Class :character   1st Qu.:56.0   Class :character
##  Median : 66.50   Mode  :character   Median :68.0   Mode  :character
##  Mean   : 66.50                      Mean   :64.6
##  3rd Qu.: 99.25                      3rd Qu.:72.0
##  Max.   :132.00                      Max.   :79.0
##     inf.mort        life.exp.f      life.exp.m        le.fm
##  Min.   :  5.00   Min.   :39.00   Min.   :38.00   Min.   :-1.000
##  1st Qu.: 14.00   1st Qu.:58.00   1st Qu.:54.75   1st Qu.: 3.000
##  Median : 36.00   Median :70.50   Median :65.00   Median : 5.000
##  Mean   : 49.65   Mean   :67.17   Mean   :62.27   Mean   : 4.902
##  3rd Qu.: 81.25   3rd Qu.:76.00   3rd Qu.:69.00   3rd Qu.: 7.000
##  Max.   :162.00   Max.   :82.00   Max.   :76.00   Max.   :11.000
```

```r
head(countries)
```

```
## # A tibble: 6 x 8
##     Row Country      life.exp agricul inf.mort life.exp.f life.exp.m le.fm
##   <int> <chr>           <int> <chr>      <int>      <int>      <int> <int>
## 1     1 Mozambique         44 64,0         162         45         43     2
## 2     2 "Etiop\xeda"       49 48,0         122         50         47     3
## 3     3 Tanzania           51 61,0          92         52         49     3
## 4     4 Sierra.Leona       43 38,0         143         45         41     4
## 5     5 Nepal              54 52,0          99         53         54    -1
## 6     6 Uganda             43 57,0         122         44         43     1
```

```r
attach(countries)
```

Since it contains some missing values we are going to omit them on their reading.

We are asked to predict the `le.fm` variable. However, looking at the summary we can see that it ranges from -1 to 11. Thus, meaning that it contains a negative value.

So a new variable, as suggested is defined such that

```
le.fm.0 <- pmax(0,le.fm)
```

What `pmax` does is return the parallel maxima and minima of the input values, which in this case it is `le.fm`. So we are forcing the minimum value -1 to be a 0, leaving the maximum as it was:

```
cat("MAXIMUM le was ", max(le.fm)," and now still is", max(le.fm.0) , "\n")
```

```
## MAXIMUM le was  11  and now still is 11
```

```
cat("MINIMUM le was ", min(le.fm)," and now is", min(le.fm.0) , "\n")
```

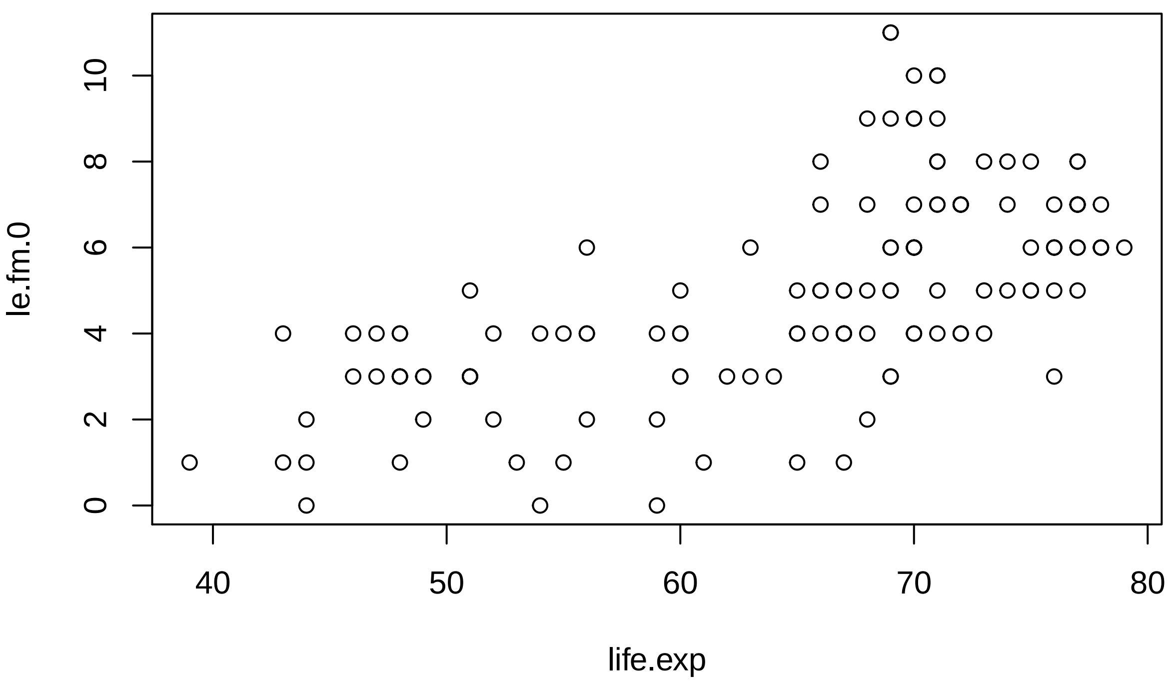
```
## MINIMUM le was  -1  and now is 0
```

Now let's proceed onto fitting a local Poisson regression using our functions onto our data.

More specifically, we are going to fit a local Poisson regression modelling our new variable `le.fm.0` as a function of `life.exp`.

This is the data we want to model:

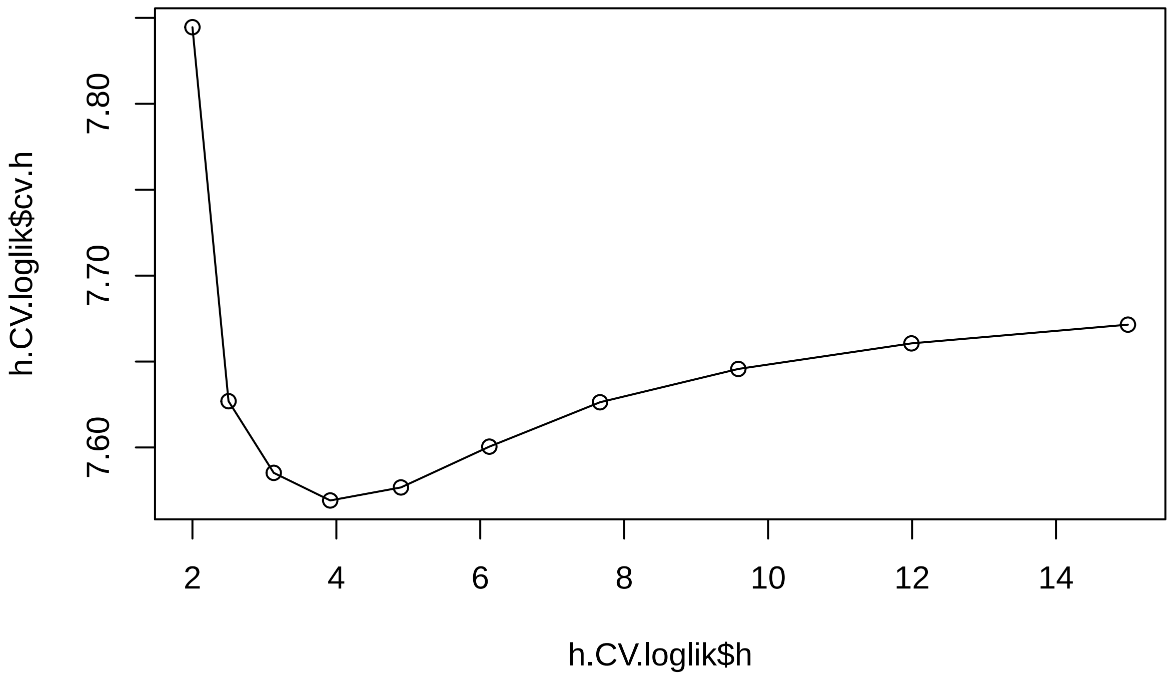
```
plot(life.exp, le.fm.0)
```



Let's first choose the best bandwidth. To preserve locality, we establish possible h that range from 2 to 15.

```
range.h = c(2, 15)
h.CV.loglik <- h.cv.sm.poisson(life.exp,le.fm.0 ,rg.h=range.h, method=loglik.CV)
```

```
plot(h.CV.loglik$h,h.CV.loglik$cv.h)
lines(h.CV.loglik$h,h.CV.loglik$cv.h)
```



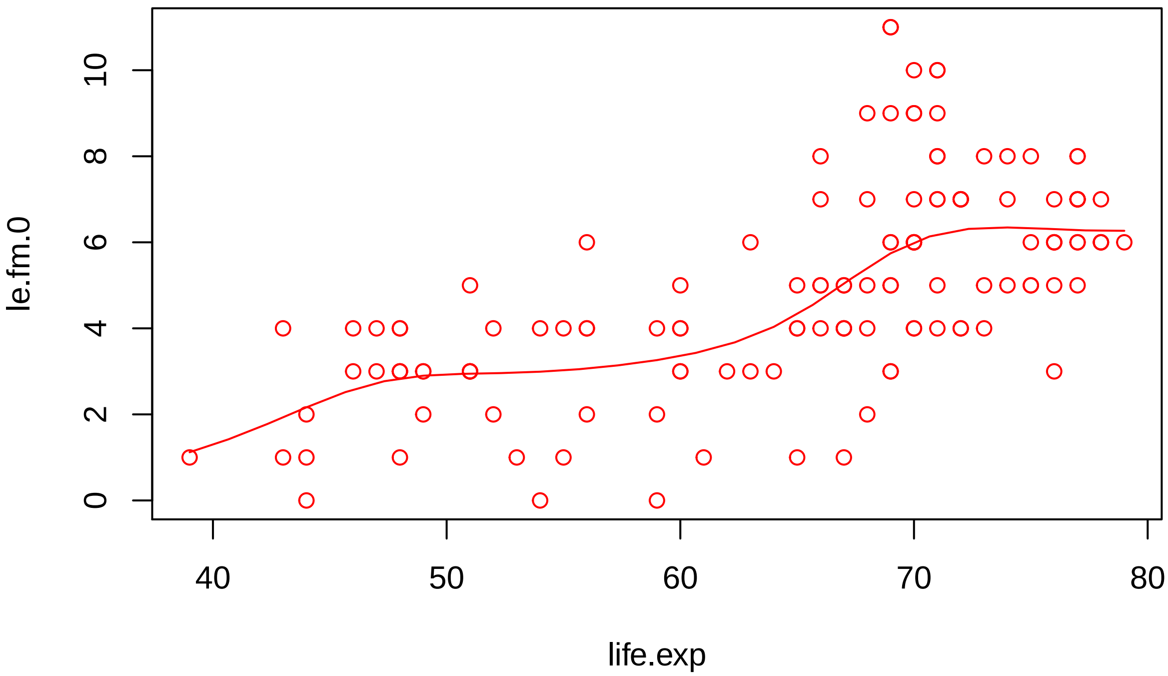

```
kable(data.frame(h.CV.loglik$h, h.CV.loglik$cv.h))
```

h.CV.loglik.h	h.CV.loglik.cv.h
2.000000	7.844568
2.501837	7.626935
3.129594	7.585231
3.914868	7.569168
4.897181	7.576745
6.125974	7.600480
7.663094	7.626313
9.585907	7.645694
11.991188	7.660572
15.000000	7.671477

We can see how the minimum log-likelihood is found close to 3, so we are going to use that value.

Let's see how our Poisson regression would be using that bandwith:

```
chosen.h =  h.CV.loglik$h.cv
m1 <- sm.poisson(life.exp, le.fm.0, h=chosen.h)
```



We can see from the plot above how the line fits our data without the line being not smooth. So the optimal bandwith is:

```
chosen.h
```

```
## [1] 3.914868
```

We can see how the regression does not go below 0, as we wanted.