Smoothing

S.Tonini, F. Chiaromonte (special thanks to J. Di Iorio and L. Insolia)

February 23th 2023

Contents

Introduction **1**
Libraries 1
Data 1

LOWESS **2**

Bin Smoothing and Kernel Smoothing **7**

General Idea on Kernel Density Estimator **11**

Introduction

Libraries

We are going to use **tidyverse** and **ggplot2**.

```
library(tidyverse)  # for data manipulation and visualization
library(ggplot2)    # for plots
```

Data

We will try to locally regress and smooth the median duration of unemployment (uempmed) based on the **economics** dataset from **ggplot2** package.

We will focus on the latest 120 months (10 years from 2005 to 2015)

```
data(economics)
help(economics)
head(economics)
```

```
## # A tibble: 6 x 6
##   date         pce    pop psavert uempmed unemploy
##   <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
## 1 1967-07-01  507. 198712    12.6     4.5     2944
```

```
## 2 1967-08-01  510. 198911    12.6     4.7     2945
## 3 1967-09-01  516. 199113    11.9     4.6     2958
## 4 1967-10-01  512. 199311    12.9     4.9     3143
## 5 1967-11-01  517. 199498    12.8     4.7     3066
## 6 1967-12-01  525. 199657    11.8     4.8     3018
```

```
dim(economics)
```

```
## [1] 574   6
```

We focus on the latest 120 months.

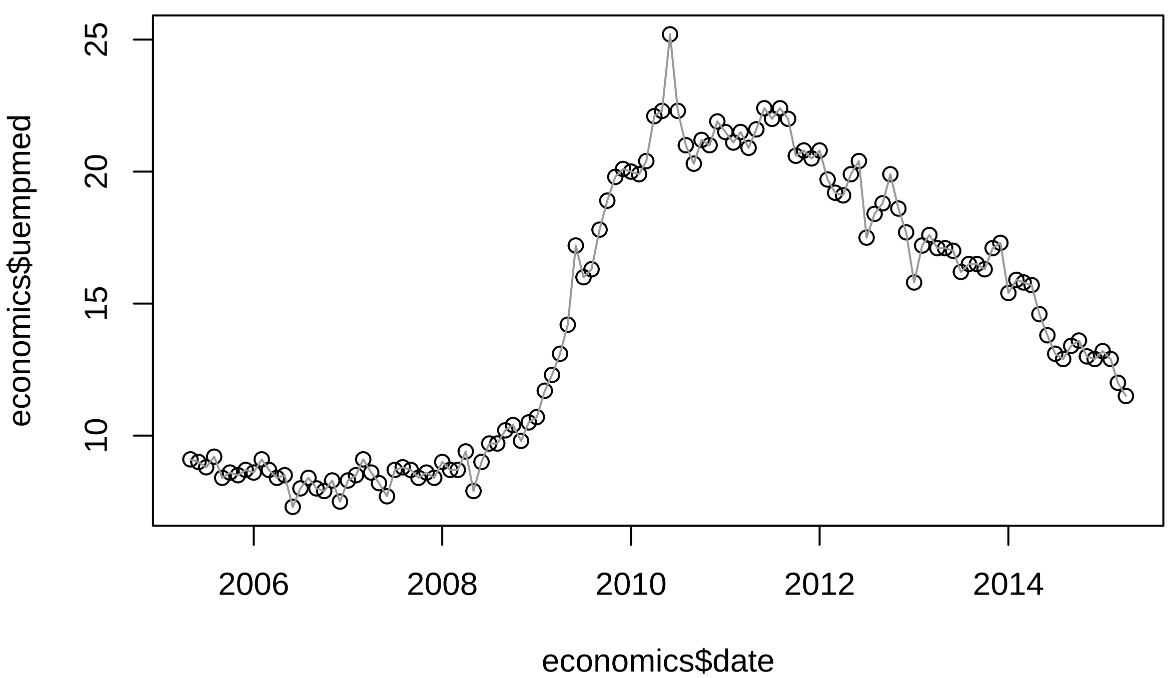
```
# first note that
dim(economics)[1] == nrow(economics)
```

```
## [1] TRUE
```

```
# subset the data
economics <- economics[(nrow(economics)-119):nrow(economics),]
dim(economics)
```

```
## [1] 120   6
```

```
plot(economics$date, economics$uempmed)
lines(economics$date, economics$uempmed, col='grey60')
```



Transform the dates into indexed from 1 (first measurement in 2005) to 120 (latest measurement in 2015).

```
economics$index <- 1:120
```

LOWESS

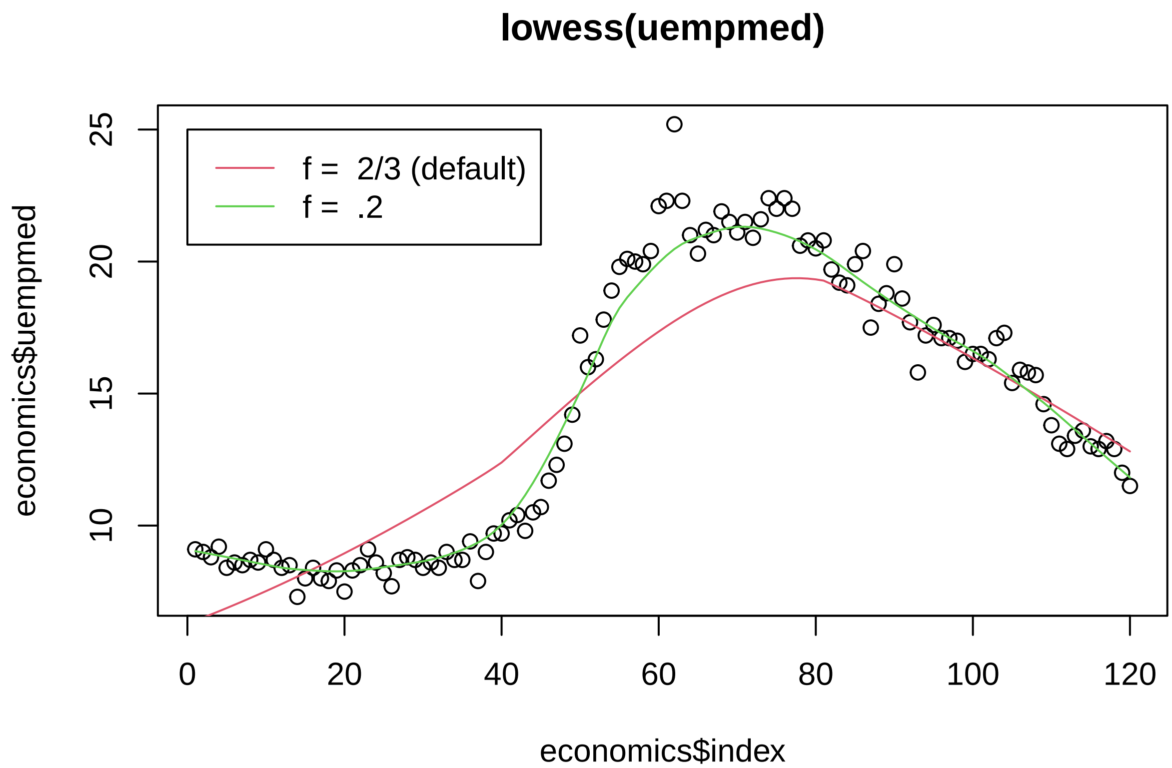
Perform LOWESS using the **lowess** function within the stats package.

It takes as inputs:

- x, y: vectors giving the coordinates of the points in the scatter plot.
- f: smoother span. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness.
- iter: the number of 'robustifying' iterations which should be performed. Using smaller values of iter will make lowess run faster.

Remark: The initial fit is done using weighted least squares. If iter > 0, further weighted fits are done using the product of the weights from the proximity of the x values and case weights derived from the residuals at the previous iteration.

```
help("lowess")

plot(economics$uempmed ~ economics$index, main = "lowess(uempmed)")
lines(lowess(economics$uempmed), col = 2)
lines(lowess(economics$uempmed, f = .2), col = 3)

legend(0, 25, c(paste("f = ", c("2/3 (default)", ".2"))), lty = 1, col = 2:3)
```



You can also perform LOWESS through the loess command in the stats package.

```
help("loess")
```

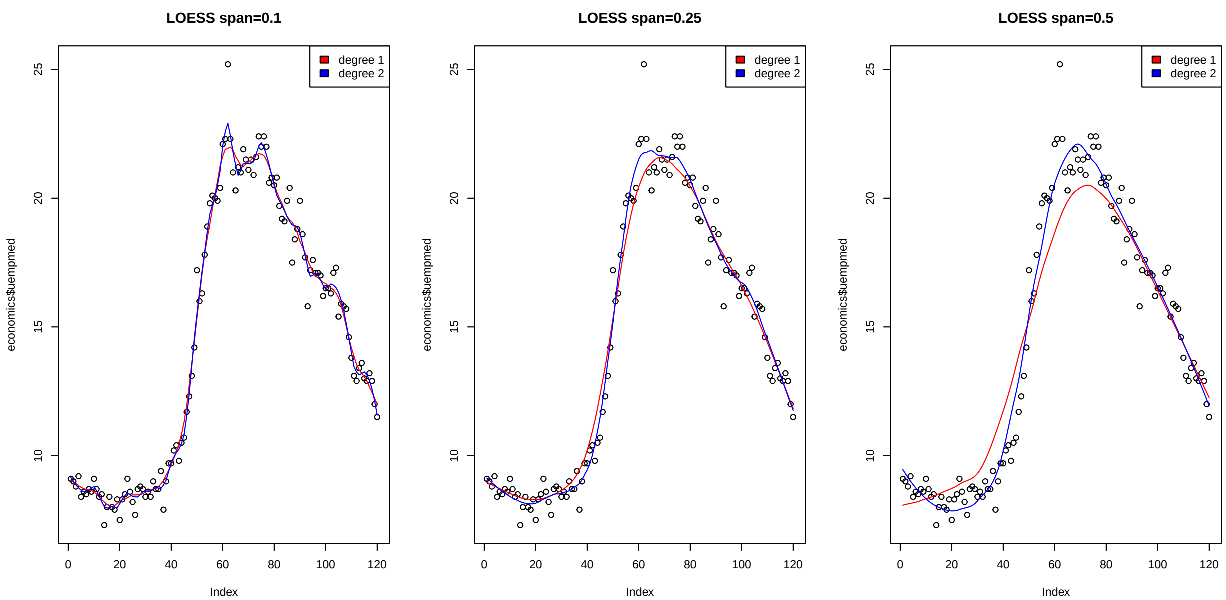
Let us focus on the following arguments:

- **formula:** a formula specifying the numeric response and one to four numeric predictors
- **data:** the dataframe
- **span:** the parameter which controls the degree of smoothing
- **degree:** the degree of the polynomials to be used, normally 1 or 2

Let's try different spans and degrees as inputs:

```
par(mfrow=c(1,3))

loess1_10 <- loess(uempmed ~ index, data = economics, span = 0.1, degree=1)
loess2_10 <- loess(uempmed ~ index, data = economics, span = 0.1, degree=2)
plot(economics$uempmed, main="LOESS span=0.1")
lines(predict(loess1_10), col='red')
lines(predict(loess2_10), col='blue')
legend("topright", fill = c("red","blue"),
       legend = c("degree 1", "degree 2"))
```

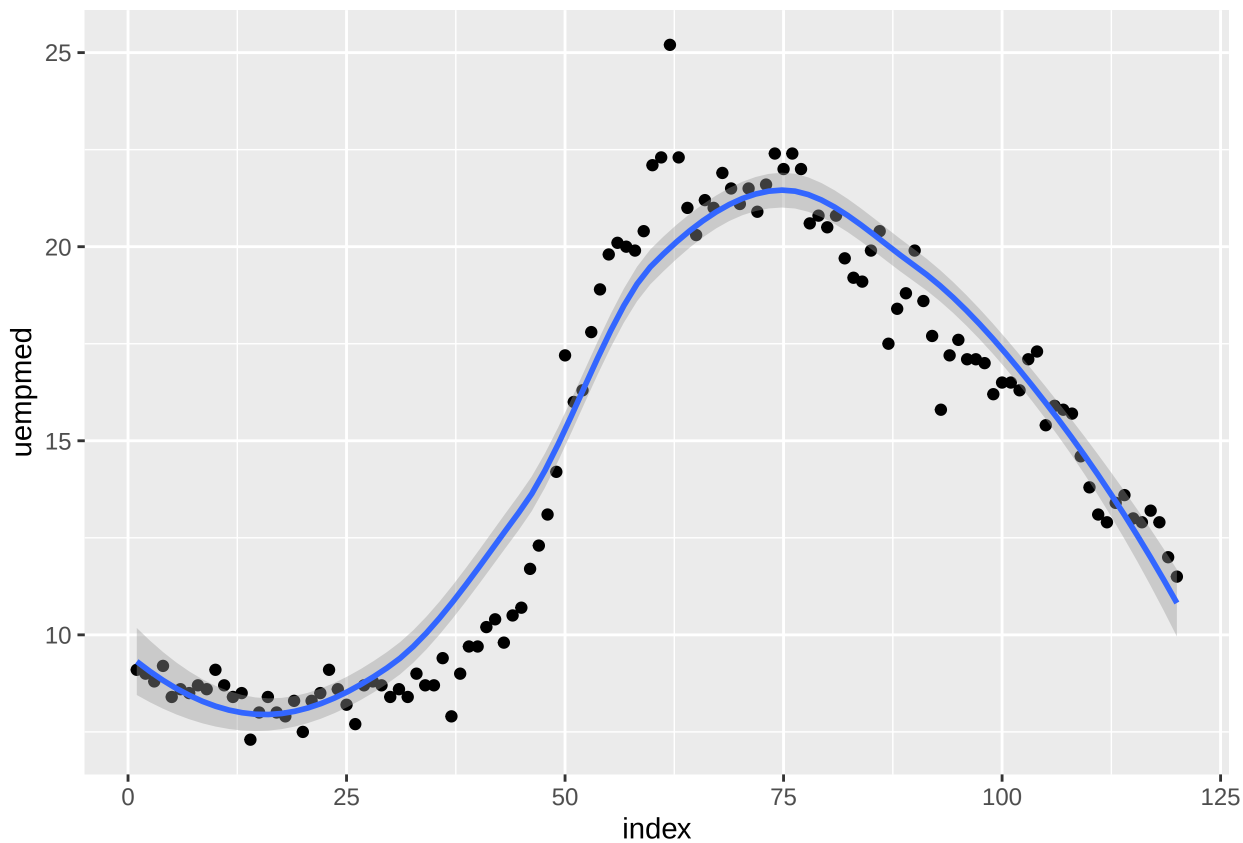
```r
# span=0.25
loess1_25 <- loess(uempmed ~ index, data = economics, span = 0.25, degree=1)
loess2_25 <- loess(uempmed ~ index, data = economics, span = 0.25, degree=2)
plot(economics$uempmed, main="LOESS span=0.25")
lines(predict(loess1_25), col='red')
lines(predict(loess2_25), col='blue')
legend("topright", fill = c("red","blue"),
       legend = c("degree 1", "degree 2"))

# span=0.5
loess1_50 <- loess(uempmed ~ index, data = economics, span = 0.5, degree=1)
loess2_50 <- loess(uempmed ~ index, data = economics, span = 0.5, degree=2)
plot(economics$uempmed, main="LOESS span=0.5")
lines(predict(loess1_50), col='red')
lines(predict(loess2_50), col='blue')
legend("topright", fill = c("red","blue"),
       legend = c("degree 1", "degree 2"))
```



The function loess has the option for fitting the local model through robust estimators similarly to lowess. See the **family** parameter for details (if “gaussian” fitting is by least-squares, and if “symmetric” a redescending estimator is used with Tukey’s biweight function).

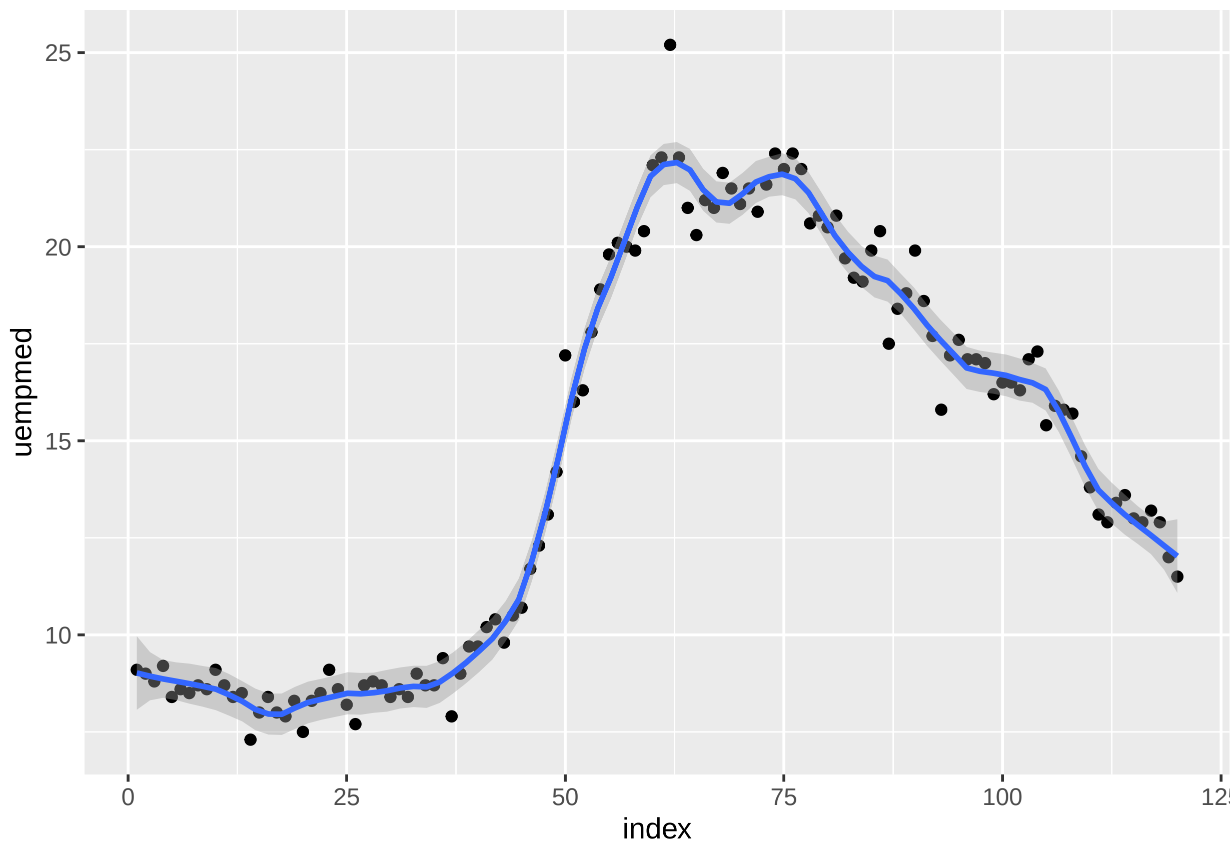
Note that ggplot uses loess in its geom_smooth function. But we should be careful for the default parameters used in there.

```r
economics %>% ggplot(aes(index,uempmed)) + geom_point() +
  geom_smooth()
```



We can change them as follows:

```
economics %>% ggplot(aes(index,uempmed)) + geom_point() +
  geom_smooth(method="loess", span=0.15, methods.args = list(degree=1))
```



Bin Smoothing and Kernel Smoothing

The general idea of smoothing is to group data points into strata in which the associated trend changes "slowly". For this reason we can assume the trend to be constant within a small window.

For our example, we will assume that unemployment remains approximately constant within a given 3-months time window.

The assumption implies that the average of the values in the window (in this case 3 months) provides a good estimate. By computing this mean for every interval (i.e. moving the window), we obtain a smooth estimate of the underlying curve.

The command that we are going to use is **ksmooth**.

```
help(ksmooth)
```

If the mean is computed giving equal weights to the points belonging to each window, we talk about "box" kernel. The result is a list with the original x and the new smoothed values y.

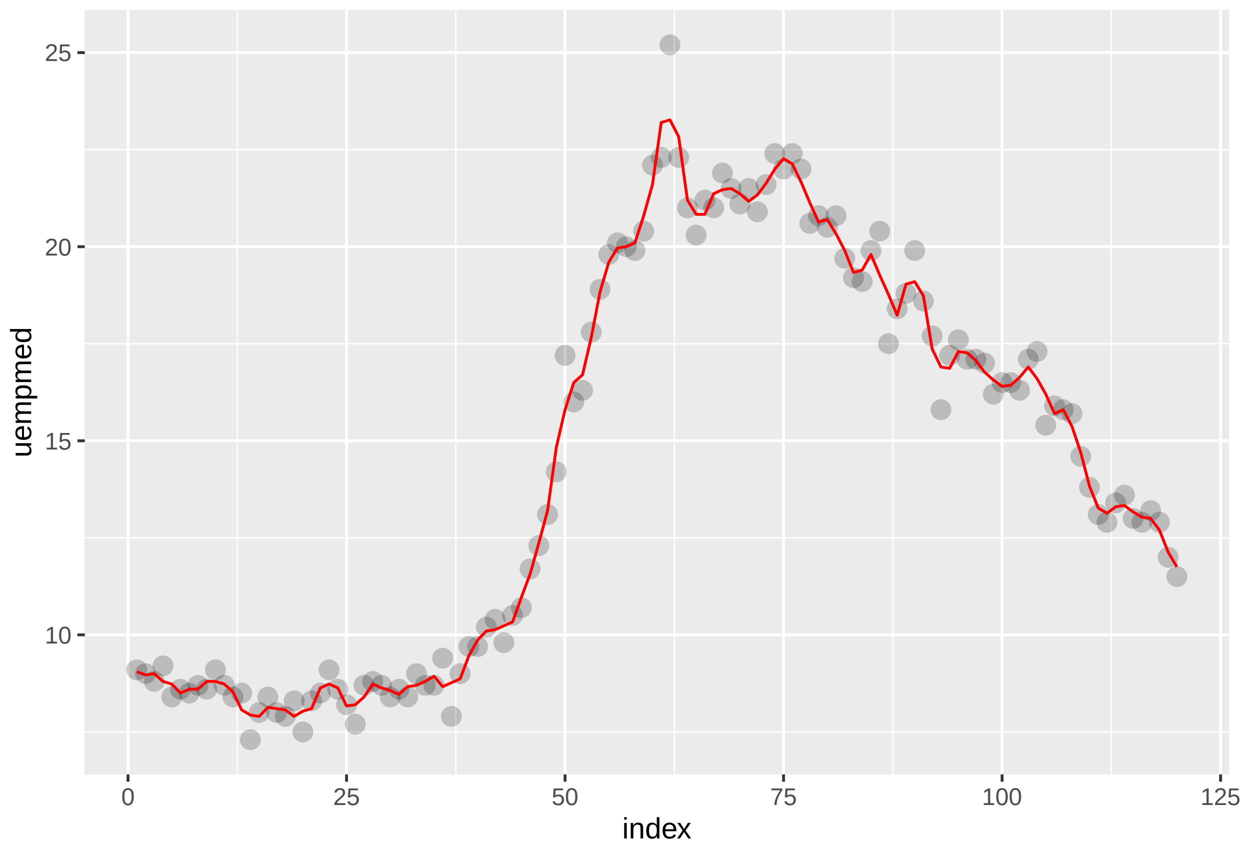
```
window <- 3
box_smooth <- ksmooth(economics$index, economics$uempmed, kernel='box', bandwidth = window)
box_smooth
```

```
## $x
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
```

```
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120
## 
## $y
##   [1]  9.050000  8.966667  9.000000  8.800000  8.733333  8.500000  8.600000
##   [8]  8.600000  8.800000  8.800000  8.733333  8.533333  8.066667  7.933333
##  [15]  7.900000  8.133333  8.100000  8.066667  7.900000  8.033333  8.100000
##  [22]  8.633333  8.733333  8.633333  8.166667  8.200000  8.400000  8.733333
##  [29]  8.633333  8.566667  8.466667  8.666667  8.700000  8.800000  8.933333
##  [36]  8.666667  8.766667  8.866667  9.466667  9.866667 10.100000 10.133333
##  [43] 10.233333 10.333333 10.966667 11.566667 12.366667 13.200000 14.833333
##  [50] 15.800000 16.500000 16.700000 17.666667 18.833333 19.600000 19.966667
##  [57] 20.000000 20.100000 20.800000 21.600000 23.200000 23.266667 22.833333
##  [64] 21.200000 20.833333 20.833333 21.366667 21.466667 21.500000 21.366667
##  [71] 21.166667 21.333333 21.633333 22.000000 22.266667 22.133333 21.666667
##  [78] 21.133333 20.633333 20.700000 20.333333 19.900000 19.333333 19.400000
##  [85] 19.800000 19.266667 18.766667 18.233333 19.033333 19.100000 18.733333
##  [92] 17.366667 16.900000 16.866667 17.300000 17.266667 17.066667 16.766667
##  [99] 16.566667 16.400000 16.433333 16.633333 16.900000 16.600000 16.200000
## [106] 15.700000 15.800000 15.366667 14.700000 13.833333 13.266667 13.133333
## [113] 13.300000 13.333333 13.166667 13.033333 13.000000 12.700000 12.133333
## [120] 11.750000
```

Let's plot our result using ggplot (unlike base R plots).

```r
economics %>% mutate(smooth = box_smooth$y) %>% ggplot(aes(index, uempmed)) +
  geom_point(size=3, alpha=0.2, color='black') + geom_line(aes(index, smooth), color='red')
```



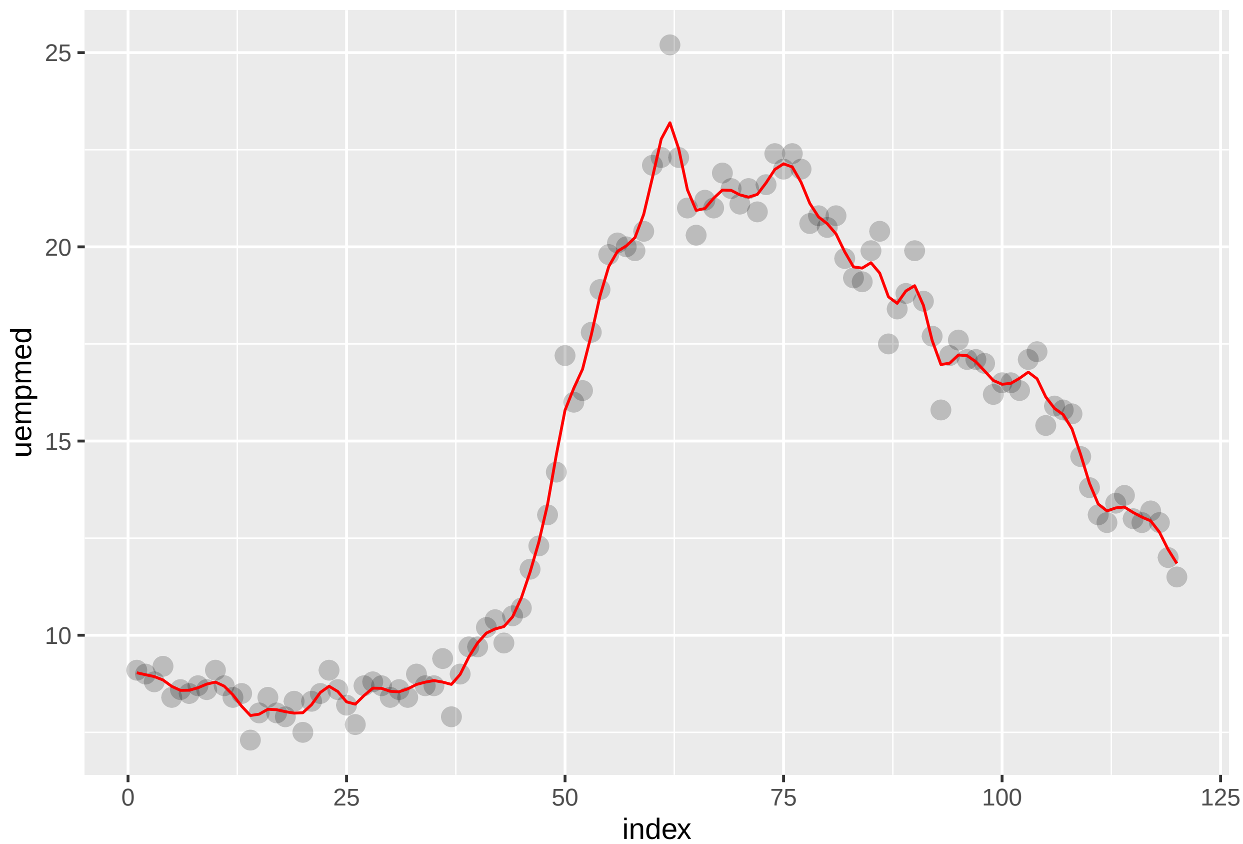
The result from the bin box smoother is quite wiggly.

The reasons for this can be the bandwidth (too small) or the uniform weights.

We can change units' weights by giving larger weights to the ones in the "central" portion of the window, thus the points at the edges will receive very little weights.

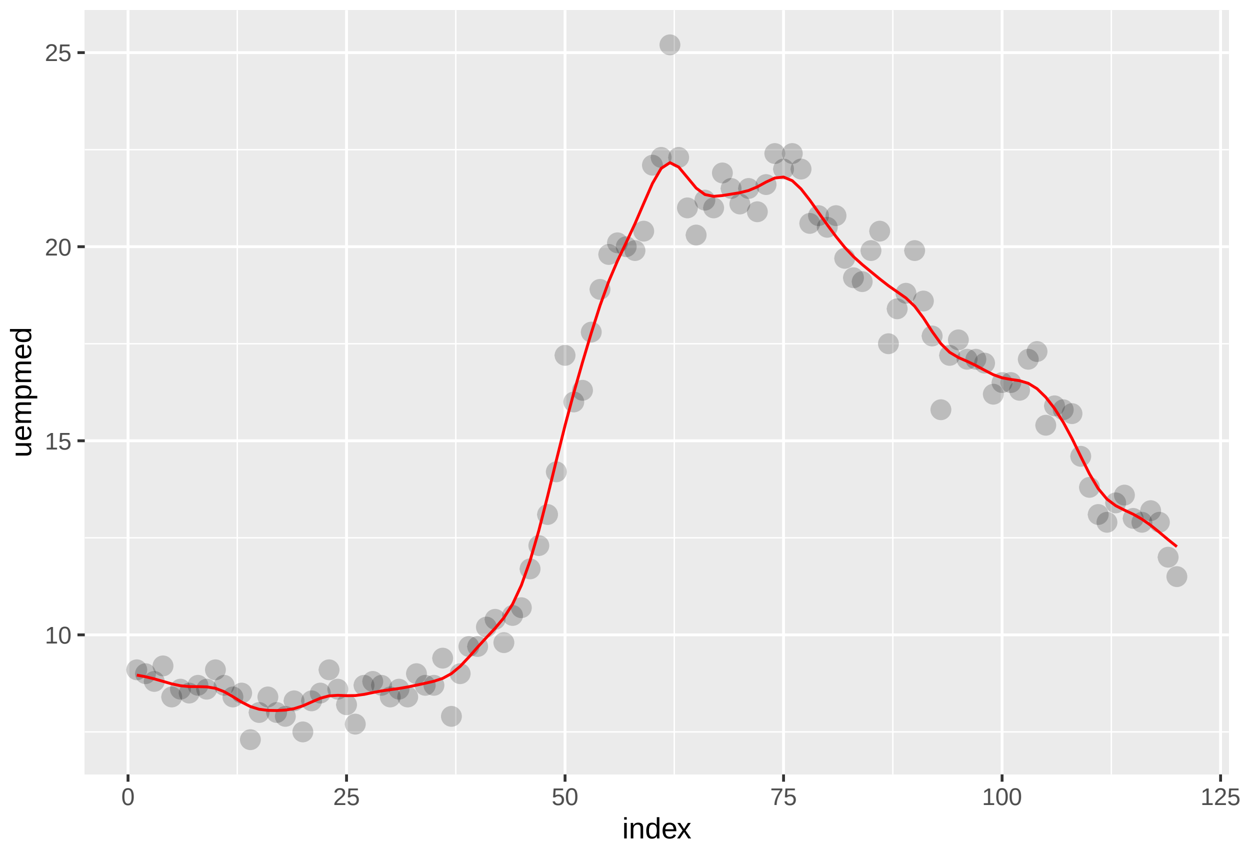
Here we will use a weighted average, where weights are provided by a normal density.

```
norm_smooth <- ksmooth(economics$index, economics$uempmed, kernel='normal', bandwidth = window)

economics %>% mutate(smooth = norm_smooth$y) %>% ggplot(aes(index, uempmed)) +
  geom_point(size=3, alpha=0.2, color='black') + geom_line(aes(index, smooth), color='red')
```



It is still wiggly! We need to change the bandwidth.

```
window <- 6 #6 month's time
norm_smooth <- ksmooth(economics$index, economics$uempmed, kernel='normal',
                       bandwidth = window)

economics %>% mutate(smooth = norm_smooth$y) %>%
  ggplot(aes(index, uempmed)) +
  geom_point(size=3, alpha=0.2, color='black') +
  geom_line(aes(index, smooth), color='red')
```



General Idea on Kernel Density Estimator

Let's simulate a new dataset, containing gender (as M/F) and weight of 400 subjects living on an undefined region of the universe.

```
set.seed(1234)
df <- data.frame(
  sex=factor(rep(c("F", "M"), each=200)),
  weight=round(c(rnorm(200, mean=55, sd=5),
                 rnorm(200, mean=65, sd=5)))
)
head(df)
```

```
##   sex weight
## 1   F     49
## 2   F     56
## 3   F     60
## 4   F     43
## 5   F     57
## 6   F     58
```

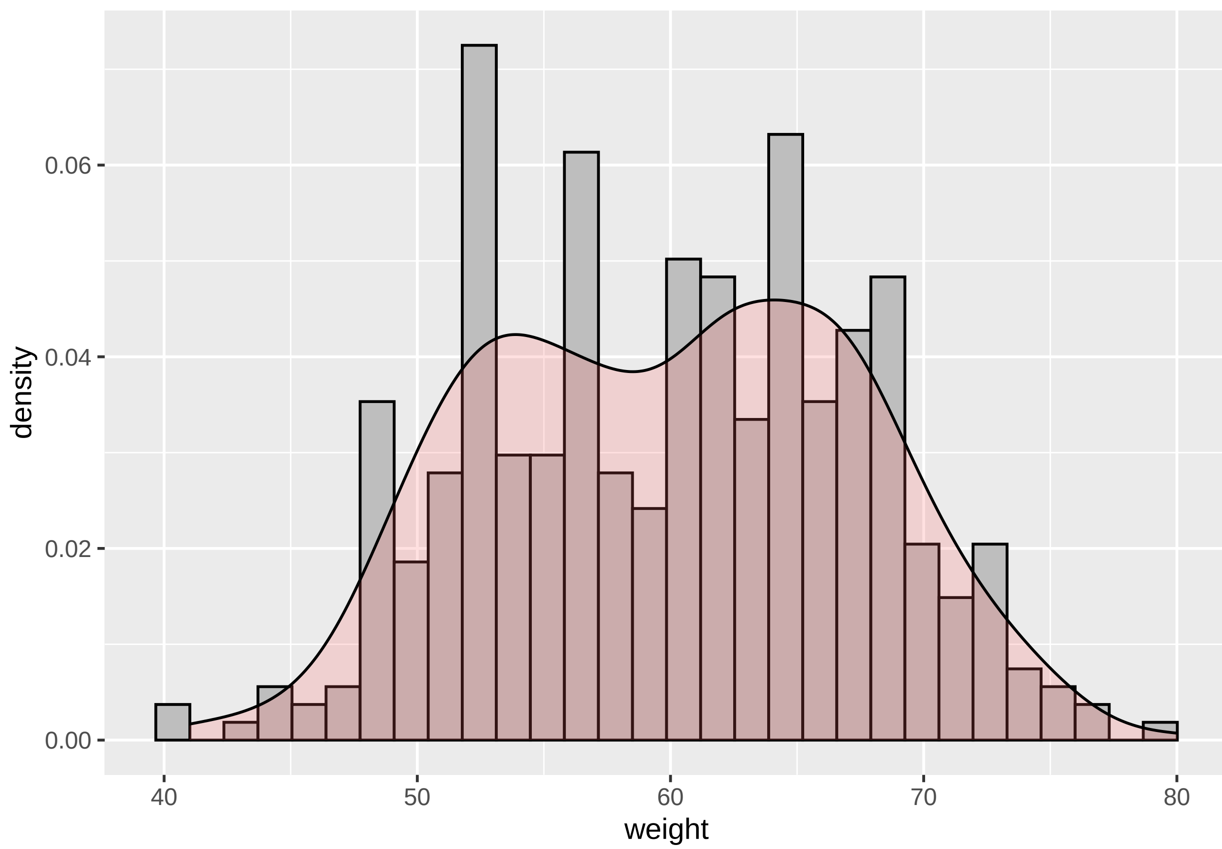
```
tail(df)
```

```
##     sex weight
```

```
## 395   M     68
## 396   M     69
## 397   M     67
## 398   M     68
## 399   M     65
## 400   M     60
```

Let's produce a histogram and its density through ggplot.

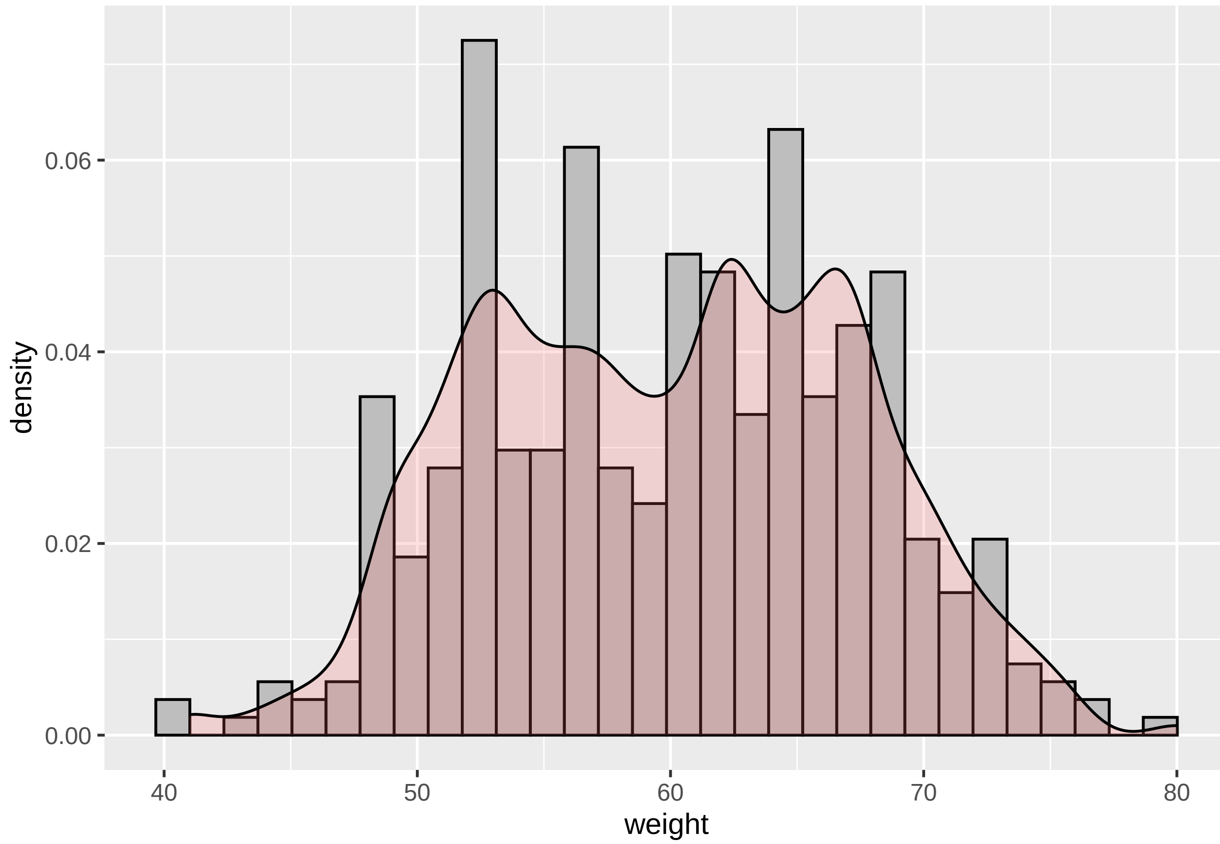
```
ggplot(df, aes(x=weight)) +
  geom_histogram(aes(y=..density..), colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666")
```



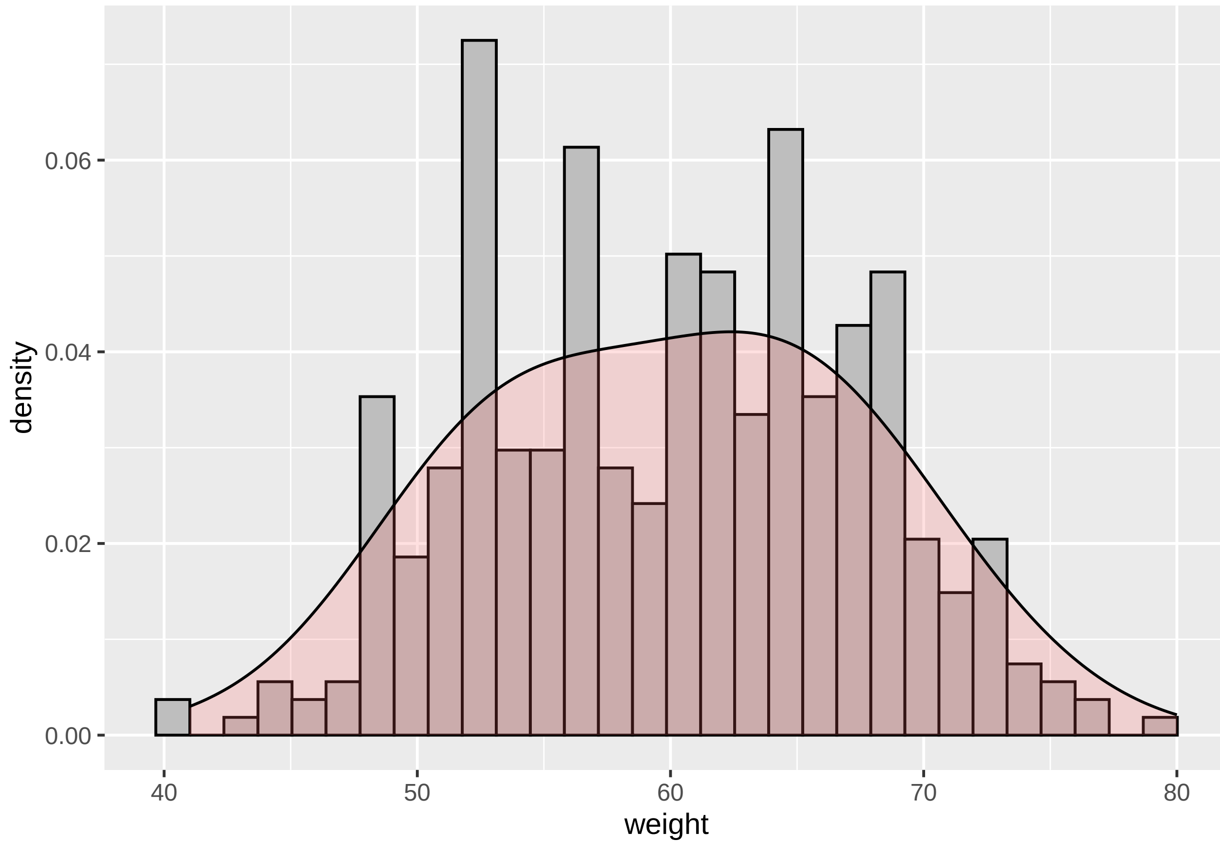
We can adjust the default density through the **adjust** argument (default is 1).

What is the adjust input used for?

```
ggplot(df, aes(x=weight)) +
  geom_histogram(aes(y=..density..), colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666", adjust=1/2)
```




```
ggplot(df, aes(x=weight)) +
  geom_histogram(aes(y=..density..), colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666", adjust=2)
```



Spoiler: it is a multiplicative bandwidth adjustment. This makes it possible to adjust the bandwidth while still using a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.