Bin Smoothing and Kernels

Machine Learning - Section 3.2.2

Marc Omar Haddad

Published: 9 March, 2020

Updated: 16 April, 2020

The General Idea of Bin Smoothing

Bin Smoothing essentially groups data points into strata in which the value of $f(x)$ can be *assumed to be constant.* The logic behind bin smoothing is that $f(x)$ changes (relatively) slowly and is thus pretty much constant within small windows of time.

For our example of the 2008 poll data (see previous notes) we can apply the assumption that **public opinion will remain approximately the same within any given week**. The result of this assumption is multiple data points with the *same expected value.*

Bin Smoothing Calculations

Setting Up Our Assumptions

Let x_0 be equal to to the precise midpoint of any given week.

$$x_0 = \text{Midpoint of week}$$

For any other day x satisfying the following condition:

$$|x - x_0| \leq 3.5$$

We assume **public opinion** $f(x)$ **to be constant**, and we will call this constant μ.

So, our assumption implies that the **expected value** of Y given X is **approximately equal to** μ, given that $|x - x_0| \leq 3.5$.

$$E[Y_i \mid X_i = x_i] \approx \mu \textbf{if } |x - x_0| \leq 3.5$$

In smoothing, the size of the interval satisfying a certain condition (in our case $|x - x_0| \leq 3.5$) is known as the **window size** or the **bandwidth** or the **span**.

Our assumptions imply that the best estimate for $f(x)$ is the average of all the Y values within our window. Let's take A_0 to be the **set of indexes** i that satisfy our condition $|x - x_0| \leq 3.5$, and N_0 to be the number (count) of indexes within A_0. Our estimate can thus be given by the following formula:

$$\hat{f}(x_0) = \frac{1}{N_0} \sum_{\substack{i \in A_0 \\ \text{in}}} Y_i$$

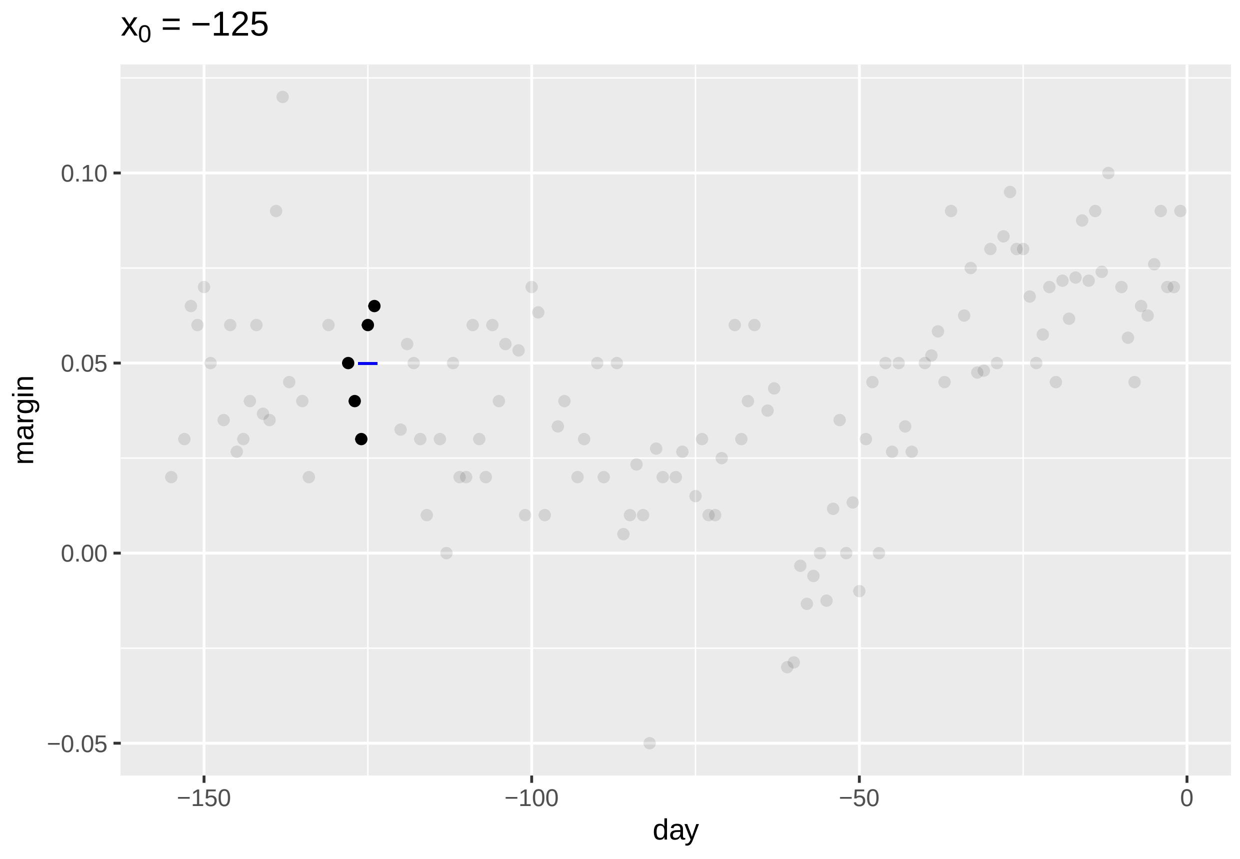
The above formula is simply **the average expected value within our window**. Dr. Rafael Irizarry states:

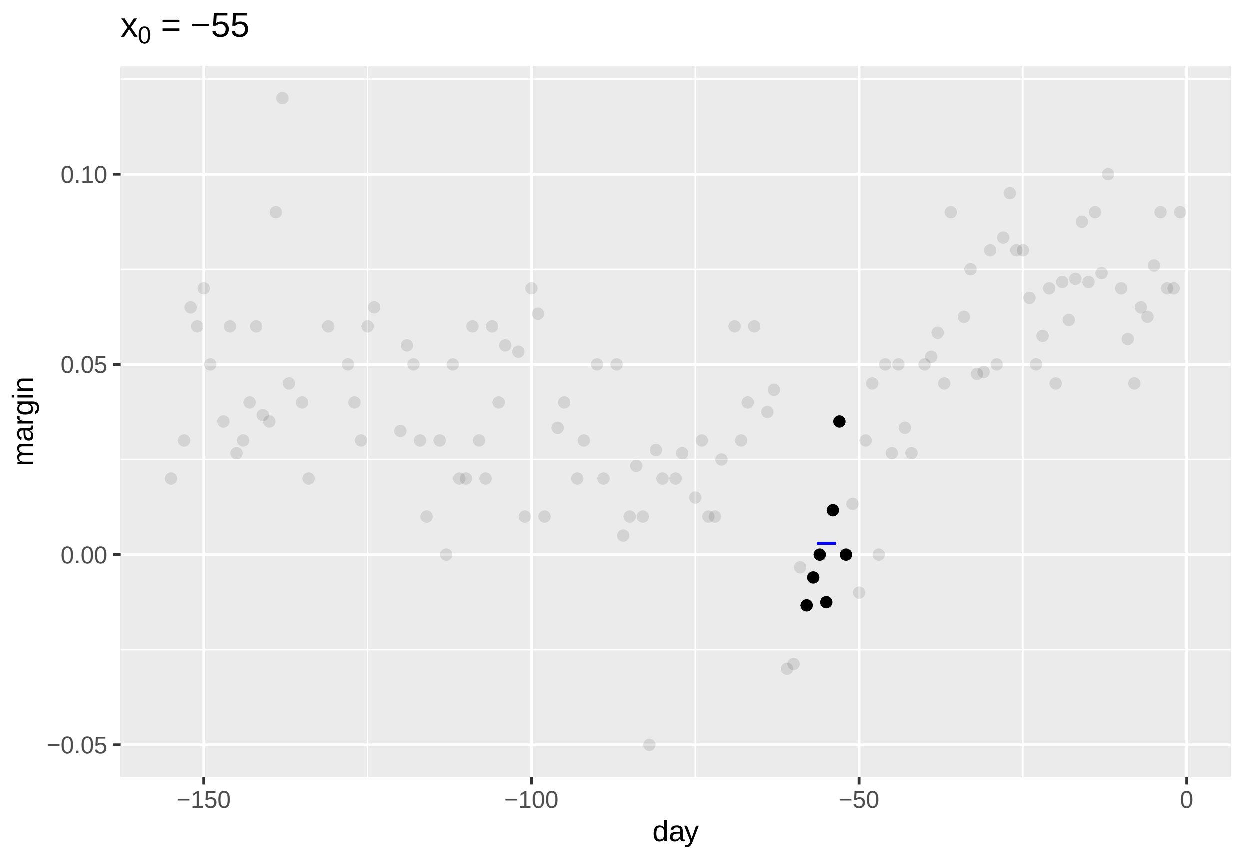
> "The idea behind bin smoothing is to make this [above] calculation for each value of x."

What this entails is making **each** value of x the center of our window and recalculating the average for all x. In our 2008 poll example, we would calculate the average of all values within a week of the day we are interested in predicting.

Examples

Let us set the midpoint of our **first example** at -125, and our **second example** at -55.



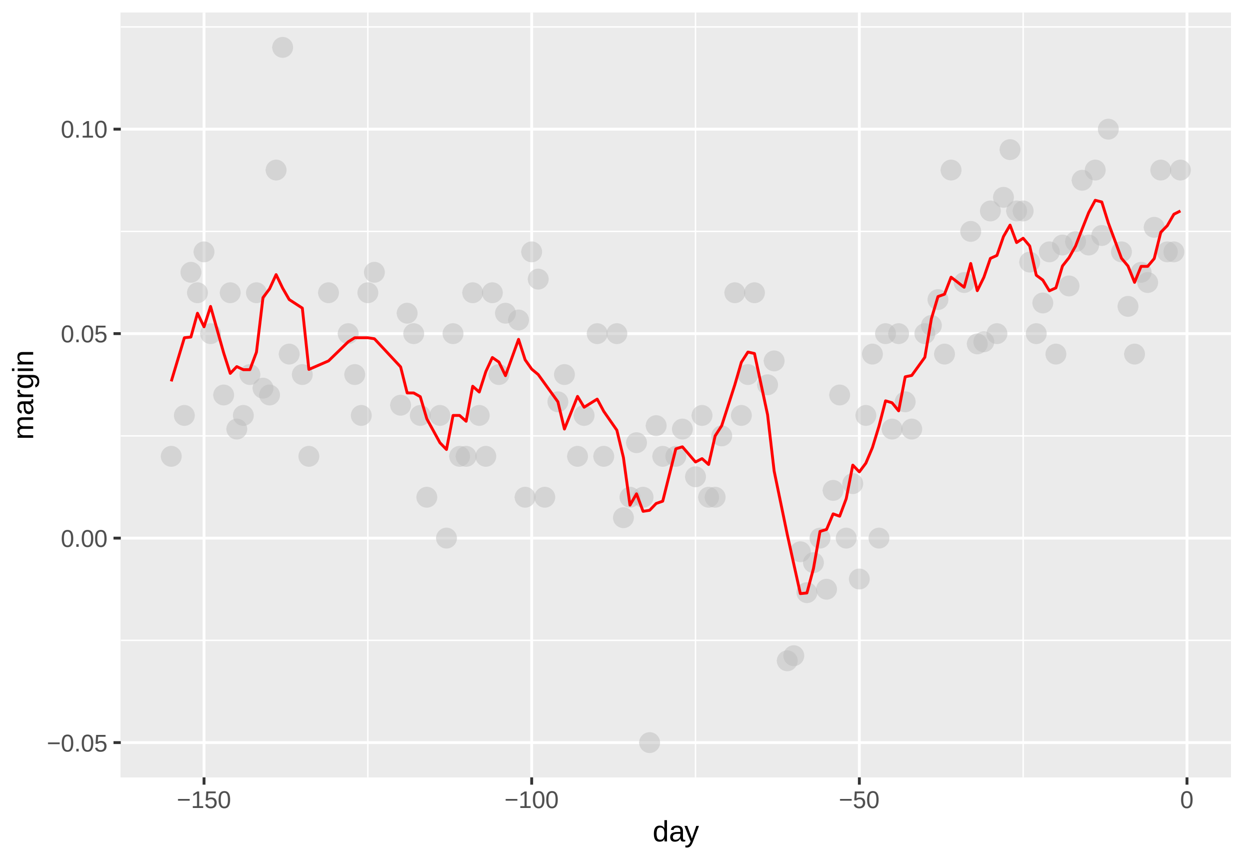


The black points are the points used to compute the average of the week containing our midpoint and the blue line is the average of those points.

First Trend Line

By repeating the exercise above for every point we can get the estimated underlying trend line for our data:

```r
span = 7
fit = with(polls_2008,
           ksmooth(day, margin, x.points = day,
                   kernel = "box", bandwidth = span)
           )
polls_2008 %>% mutate(smooth = fit$y) %>%
  ggplot(aes(day, margin)) +
  geom_point(size = 3, alpha = 0.5, color = "grey") +
  geom_line(aes(day, smooth), color = "red")
```



As you can see from the above plot, our “smooth” line is actually quite “wiggly”. The reason for this is the fact that for every step within our window (7 days) **2 points change**, which is a substantial percentage of the window and can lead to volatile trend lines.

We can mitigate this effect by **taking weighted averages that give the center points more weight**. This would mean that the 2 points at the edges of our windows would receive the least weight. The functions with which we compute the aforementioned weights are known as the **Kernel**.

The Kernel

Mathematical Representation

Any time we use a bin smoother, we can consider it to be an approach that uses a kernel. Mathematically, the weighting within a kernel can be expressed like so:

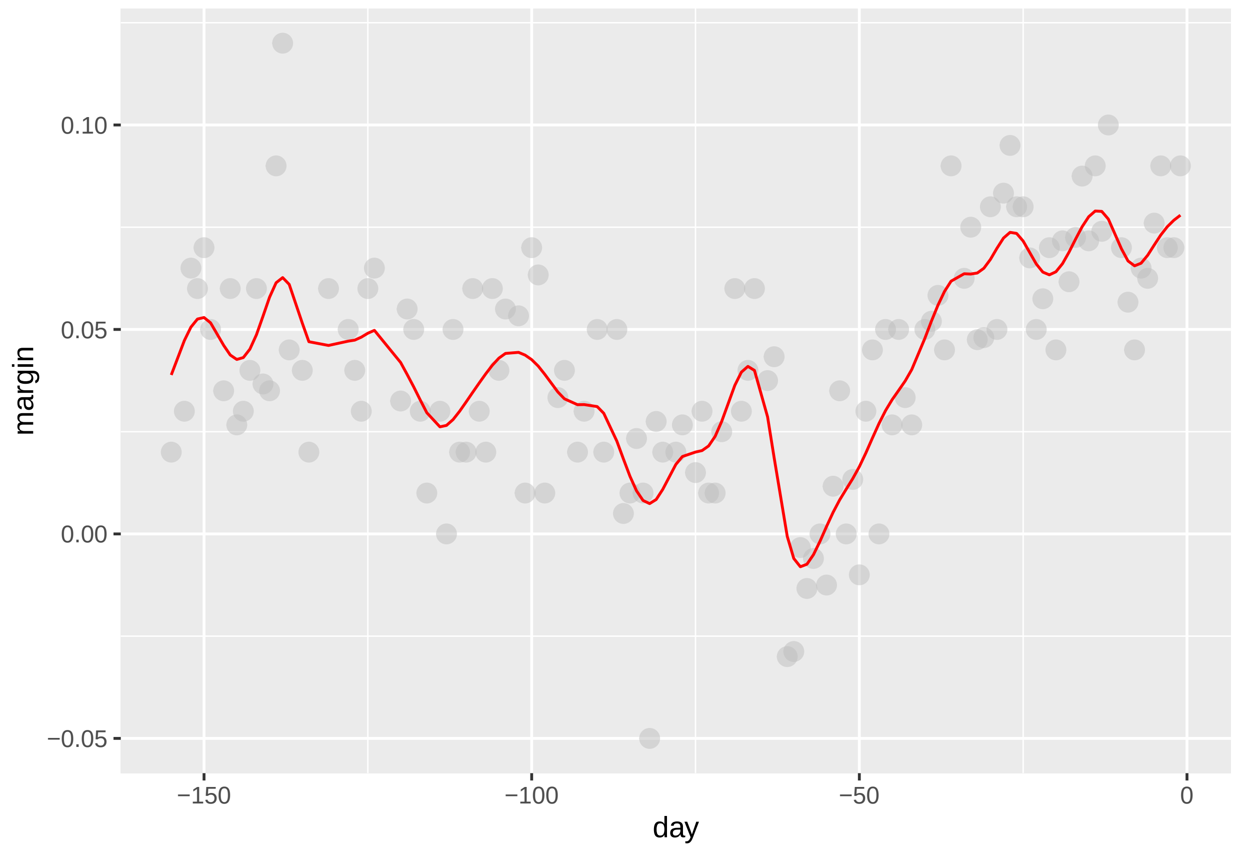
$$\hat{f}(x_0) = \sum_{i=1}^{N} w_0(x_i) Y_i$$

Each point receives a weight (w) between 0, for points that are outside of the window, and $\frac{1}{N_0}$, for points within the window, with N_0 being the number of points within that week.

Second Trend Line

We can achieve a smoother curve by applying a **Normal/Gaussian Density** when assigning weights. In the `ksmooth` function in the previous plot, instead of setting our `kernel` to `"box"` we will now attempt to set it to "normal".

```
span = 7
fit = with(polls_2008,
           ksmooth(day, margin, x.points = day,
                   kernel = "normal", bandwidth = span)
           )
polls_2008 %>% mutate(smooth = fit$y) %>%
  ggplot(aes(day, margin)) +
  geom_point(size = 3, alpha = 0.5, color = "grey") +
  geom_line(aes(day, smooth), color = "red")
```



Our second plot is considerably smoother than our first. However, Dr. Irizarry warns:

> "… in practice, we typically prefer methods that use slightly more complex models than fitting a constant."

In the next chapter, we will learn new methods to improve our smoothing.