Local Weighted Regression (Loess)

Machine Learning - Section 3.2.3

Marc Omar Haddad

Published: 16 April, 2020

Updated: 25 April, 2020

Limitation of Bin Smoother Approach

A limitation of the earlier bin smoother approach (see previous notes) was that we were required to take very small windows for the approximately constant assumption to hold. This results in a small number of data points to average.

Local Weighted Regression (AKA Loess) permits us to consider larger windows and, thus, achieve more accurate, smoother trend lines.

Mathematics of Loess

Taylor's Theorem

If you look close enough at any smooth function $f(x)$, the relationship will appear **linear**. The previous bin smoother approach assumed that our function is approximately constant within a window, whereas now we shall assume the function is locally linear. This permits us to use larger window sizes without sacrificing accuracy.

Formula

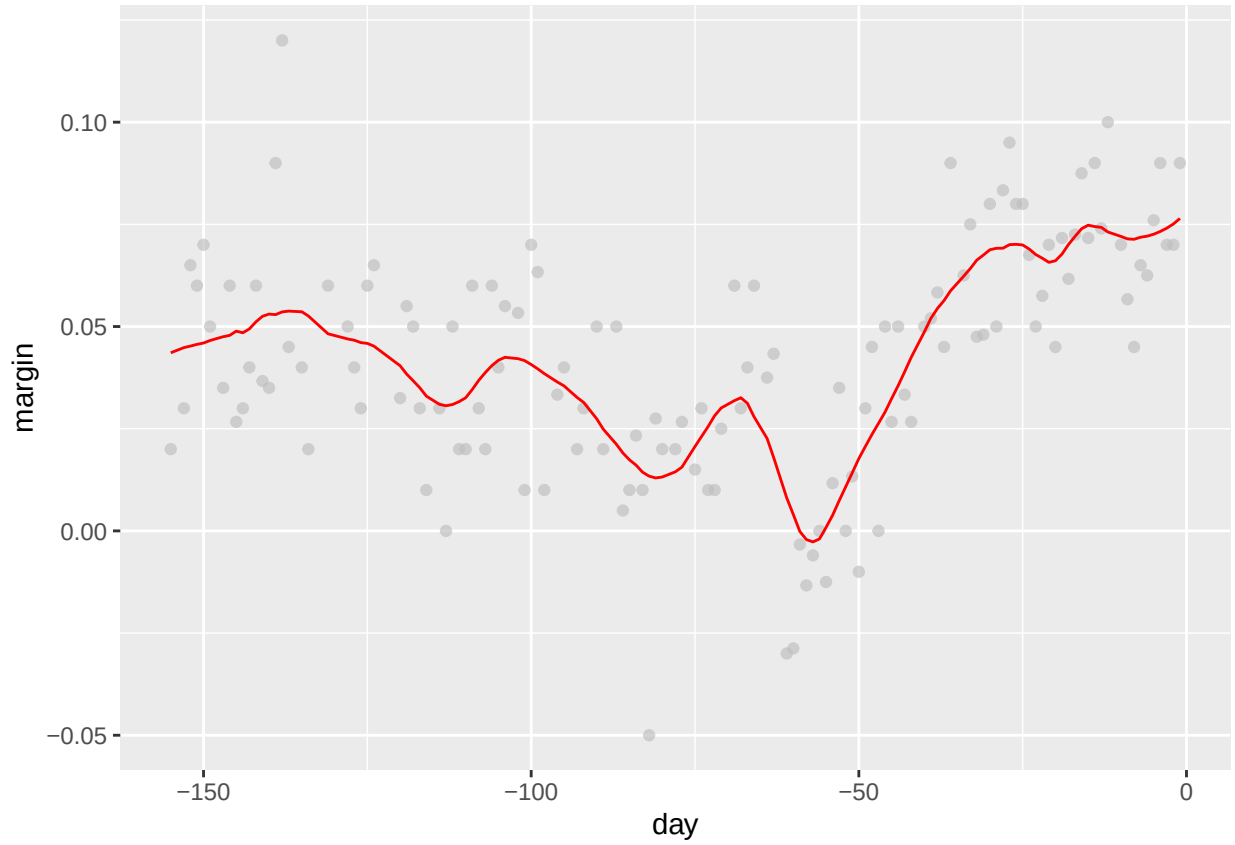
For our example we will start by using a 3 **week window**. Mathematically, our line equation for **one** window is calculated as:

$$E\left[Y_i \mid X_i = x_i\right] = \beta_0 + \beta_1(x_i - x_0)$$
$$\textbf{if } |x_i - x_0| \leq 10.5$$

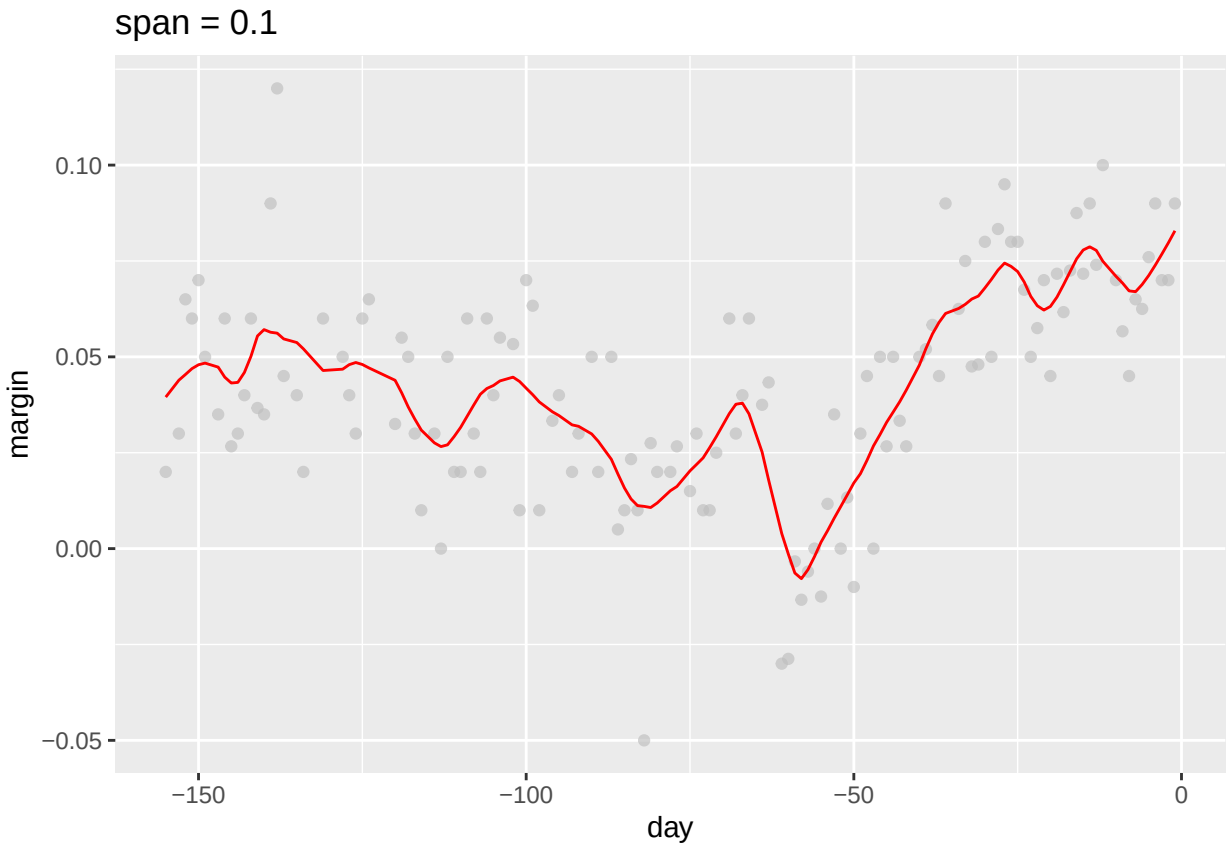
We assume Y given X to be a line within that window (as evidenced by the linear equation).

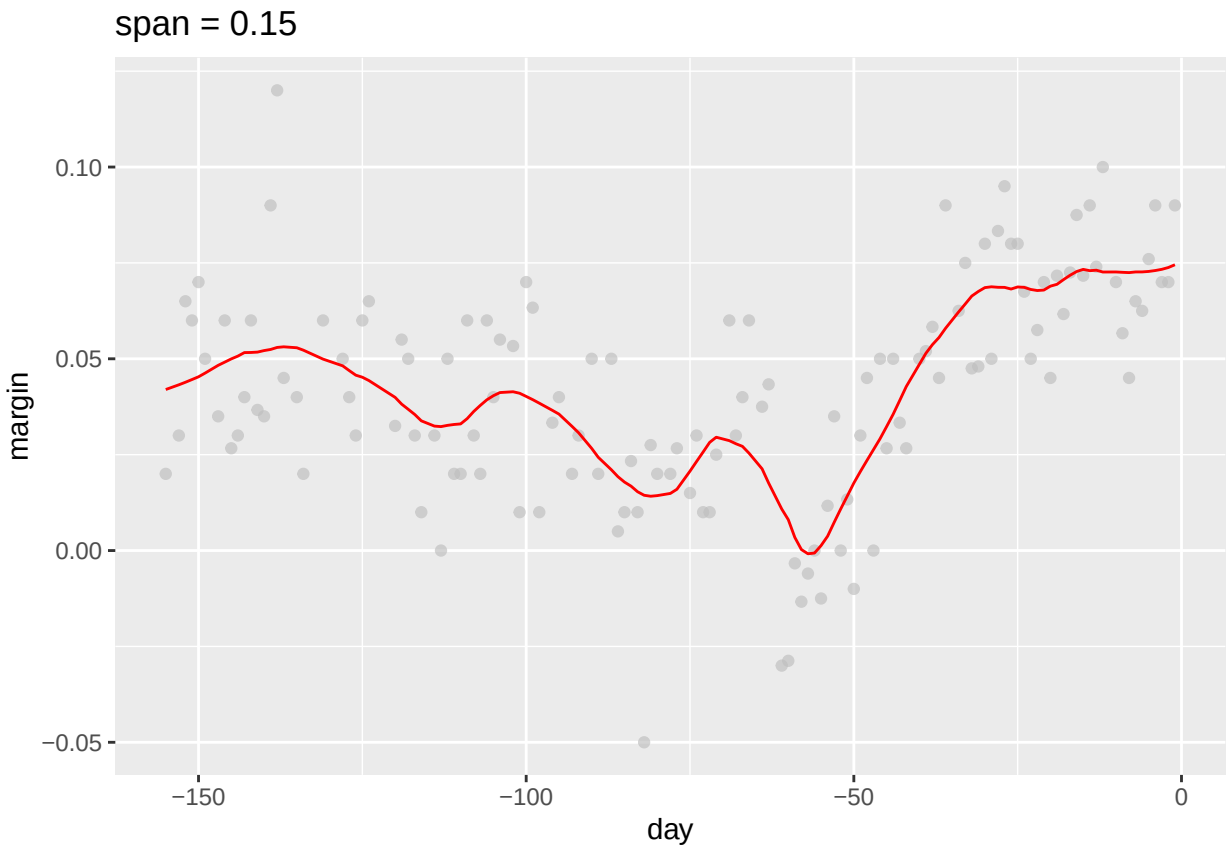
Plots

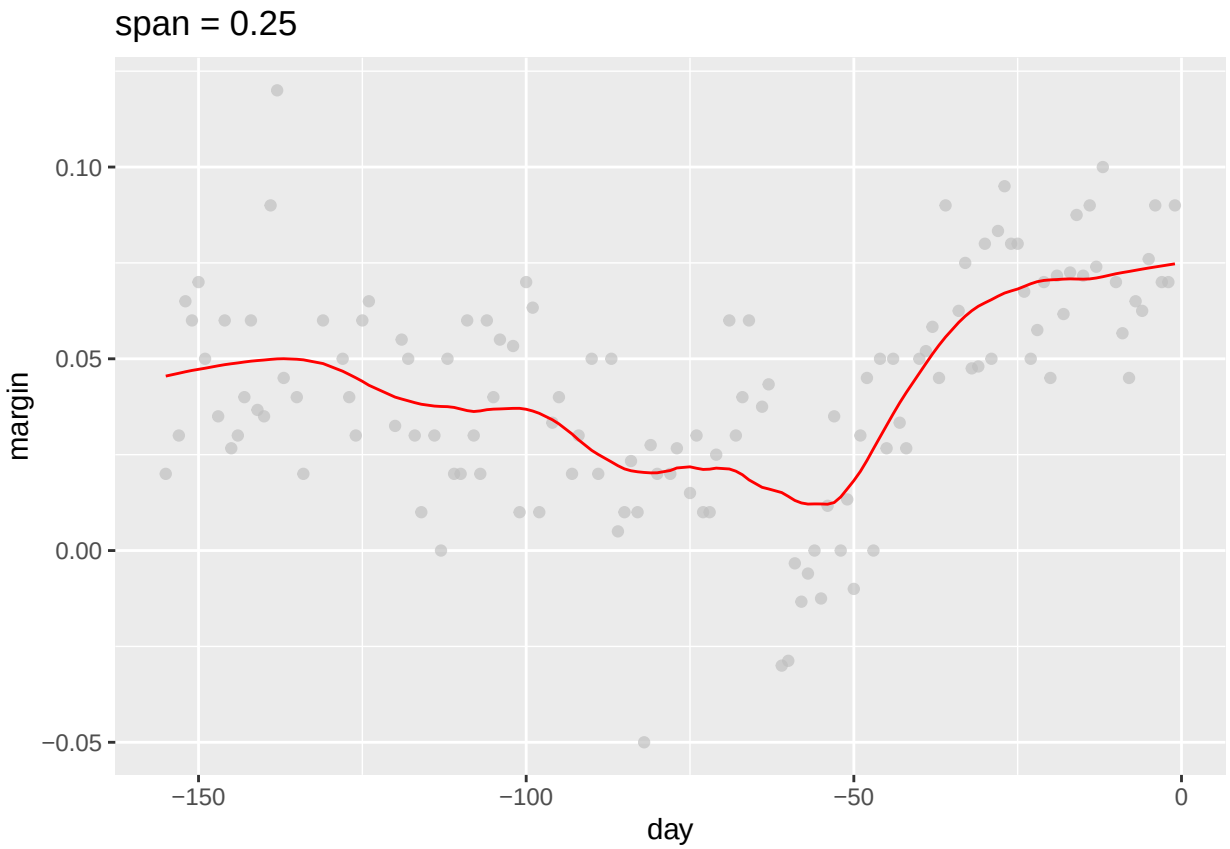
```
total_days = diff(range(polls_2008$day))
span = 21/total_days

fit = loess(margin ~ day, degree = 1, span = span, data = polls_2008)

polls_2008 %>%
  mutate(smooth = fit$fitted) %>%
  ggplot(aes(day, margin)) +
  geom_point(alpha = 0.65, color = "grey") +
  geom_line(aes(day, smooth), color = "red")
```

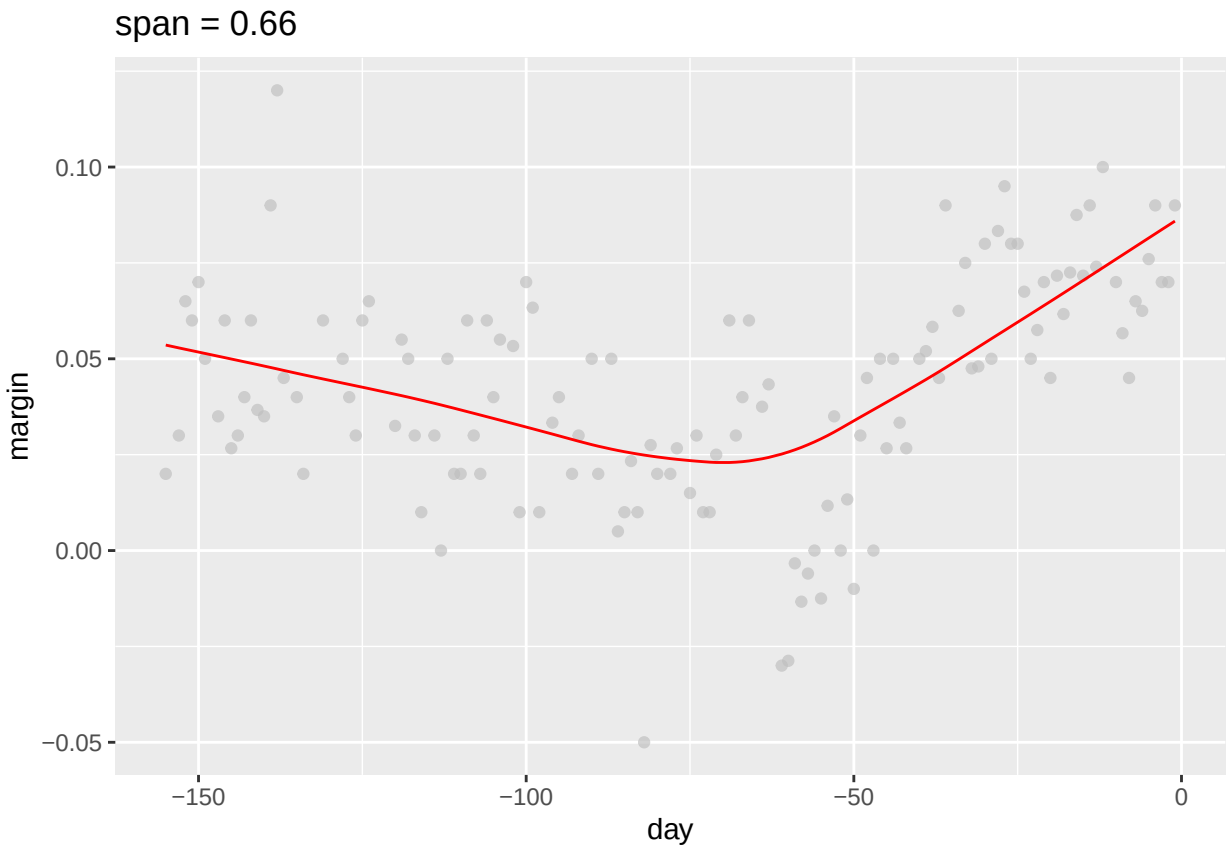


Different `span` would give us different estimates.




span = 0.15
0.10
0.05
0.00
−0.05
margin
−150
−100
−50
0
day


span = 0.25
0.10
0.05
0.00
−0.05
margin
−150
−100
−50
0
day



Three Differences Between Bin Smoothing and Loess

The First Difference:
Rather than keeping the bin sizes the same (regardless of number of points within bin), **Loess keeps the number of points used in the local fit the same**. The width of this window is controlled by the `span` argument, which expects a proportion. For example: if N is the number of total data points, and our `span` is equal to 0.5, **the number of data points within *any* given window will always be** $0.5 \times N$ **closest points to the fit**.

The Second Difference:
When fitting a line locally, **Loess uses a weighted approach**. What this means is that instead of minimizing the Least Squares, **we must minimize a weighted version of Least Squares**. Represented mathematically like so:

$$\sum_{i=1}^{N} w_0(x_i)[Y_i - \{\beta_0 + \beta_1(x_i - x_0)\}]^2$$

Instead of using the Gaussian (normal) kernel for weighting, **Loess uses the Tukey Tri-Weight function**, expressed like this:

$$W(u) = (1 - |u|^3)^3 \quad \textbf{if } |u| \leq 1$$
$$\textbf{and}$$
$$W(u) = 0 \quad \textbf{if } |u| > 1$$

And we use the following formula to actually define the weights:

$$w_0(x_i) = W\Big(\frac{x_i - x_0}{h}\Big)$$

The Third Difference:
Loess offers the option to **fit the local model robustly**. What this means:

> "An iterative algorithm is implemented in which, after fitting a model in one iteration, **outliers** are detected and **down-weighted** for the next iteration." - Dr. Rafael Irizarry

To use this robust option in `R`, we must set the `family` argument to `"symmetric"`.

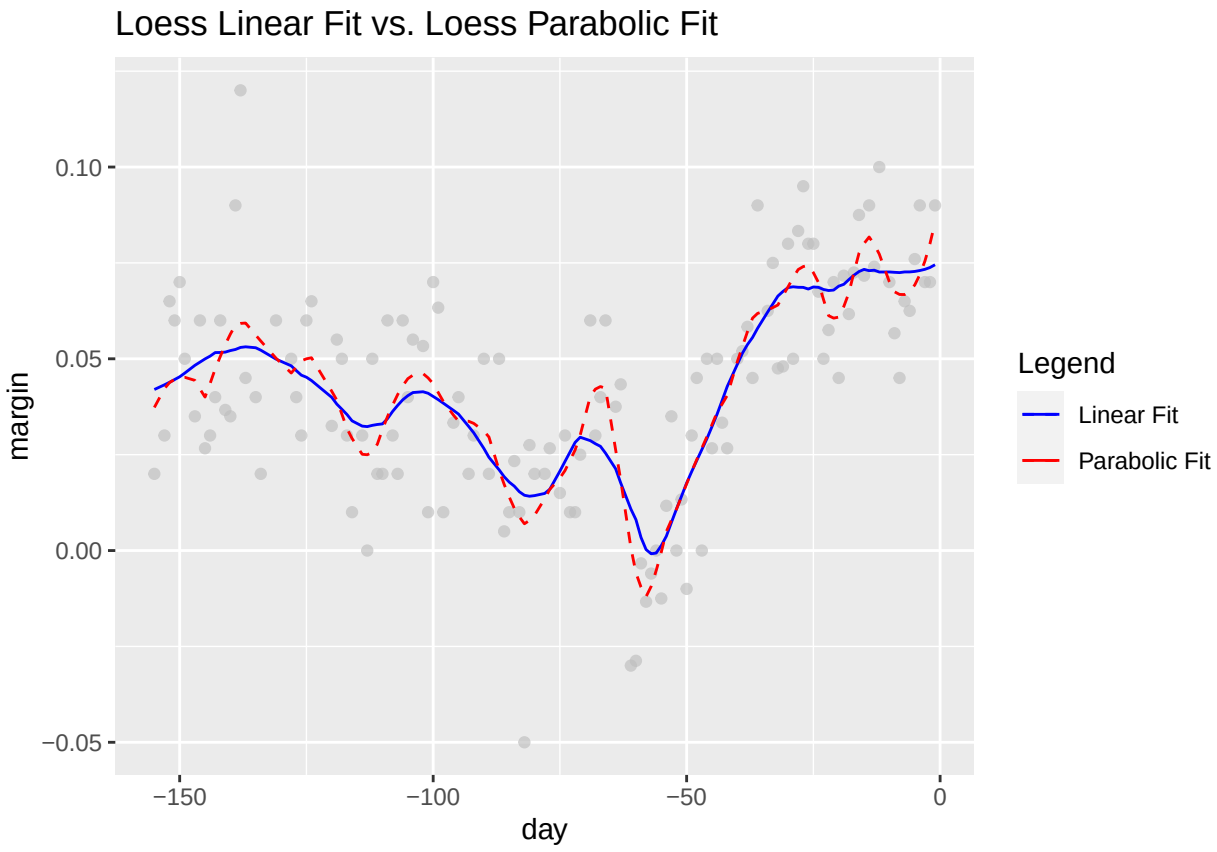
Important Note on Default Loess Behavior

Taylor's Theorem also states that if you look closely at any smooth function $f(x)$, but not close enough to see a linear relationship, you will see a **parabolic relationship**. This allows for **bigger local windows** since you can capture more of the data without the function deviating from a parabolic shape. Thus, instead of a linear local model, the local model will look like this:

$$E\left[Y_i \mid X_i = x_i\right] = \beta_0 + \beta_1(x_i - x_0) + \beta_2(x_i - x_0)^2$$
$$\textbf{if } |x_i - x_0| \leq h$$

The above is actually the default procedure when using the `loess()` function. However, in our plotted curves earlier, we set the `degree` value equal to 1. This overrides the default behavior of looking at parabolic local modes (i.e. `degree = 2`), and instead plots according to linear local modes (i.e. `degree = 1`). We do this because **a linear local method appears to give us a smoother curve**, as seen here:

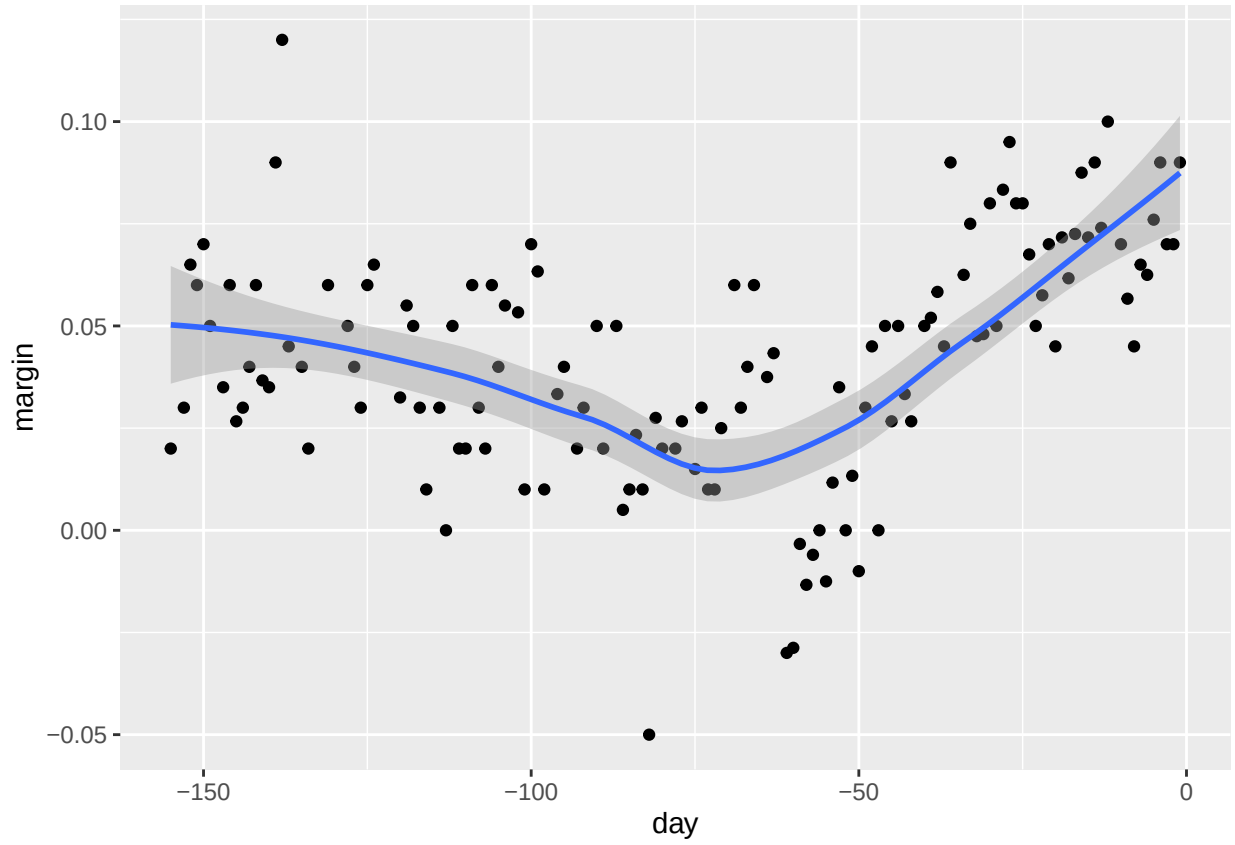
```
fit_linear = loess(margin ~ day, degree = 1,
                   span = 0.15, data = polls_2008)
fit_parabola = loess(margin ~ day, span = 0.15, data = polls_2008)
colors = c("Linear Fit" = "blue", "Parabolic Fit" = "red")
polls_2008 %>%
  mutate(smooth_linear = fit_linear$fitted,
         smooth_parabola = fit_parabola$fitted) %>%
  ggplot(aes(x = day, y = margin)) +
  geom_point(alpha = 0.65, color = "grey") +
  geom_line(aes(day, smooth_linear, color = "Linear Fit")) +
  geom_line(aes(day, smooth_parabola, color = "Parabolic Fit"),
            linetype = "dashed") +
  labs(title = "Loess Linear Fit vs. Loess Parabolic Fit",
       color = "Legend") +
  scale_color_manual(values = colors)
```



As you can see in our above plot, the linear fit appears smoother than the parabolic fit; thus, we choose to set `degree` equal to 1.

The ggplot function geom_smooth() uses Loess and simplifies the process of fitting a line.

```
polls_2008 %>%
  ggplot(aes(day, margin)) +
  geom_point() +
  geom_smooth()
```



However, the default behavior doesn't necessarily give us the optimal fit. We can tweak the fitting process by changing the `span` and `method.args` arguments:

```
polls_2008 %>%
  ggplot(aes(day, margin)) +
  geom_point() +
  geom_smooth(span = 0.15, method = "loess", method.args = list(degree = 1))
```