Smoothing

EPsy 8264

2018-06-06

The data *duff.csv* contains quarterly data on the total revenues for the Duff Brewing Company.

```
# Import data
duff = read_csv("data/duff.csv")
duff
```

```
## # A tibble: 25 x 2
##    quarter  revenue
##      <int>    <dbl>
##  1       1 2962901.
##  2       2 1875403.
##  3       3 6569281.
##  4       4 2780925.
##  5       5 1622039.
##  6       6 3853289.
##  7       7 3950326.
##  8       8 3846413.
##  9       9 4188955.
## 10      10 2674918.
## # ... with 15 more rows
```

Below we plot the revenue versus quarter. In the figure we see that there appears to be a general tendency for the revenues to increase over time. Secondly, we see that revenues seems to be cyclic over a year—the trend within a year is similar across years. For example, revenues seem to peak in the third quarter in most years and then diminish in the fourth quarter of the year.

```
ggplot(data = duff, aes(x = quarter, y = revenue)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  scale_x_continuous(name = "Quarter", breaks = seq(from = 0, to = 28, by = 4)) +
  ylab("Revenue")
```

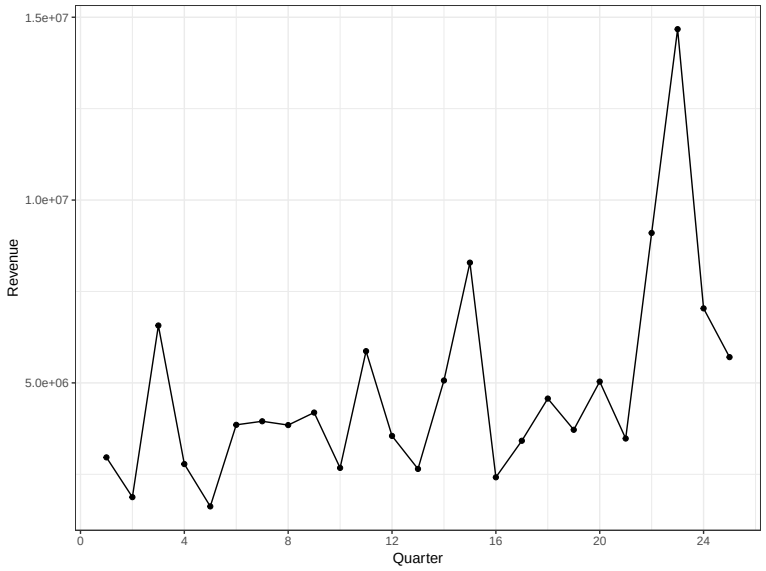

Figure 1: *Revenue by quarter for Duff Brewing Company*

The within-year cycles/trend can make detecting the overall trend in the data more difficult. One statistical method for helping to detect trands in noisy data is *smoothing*. Smoothing methods are a family of forecasting/prediction methods that tend to average values over multiple periods/neighborhoods in the data. We will examine several smoothing methods in this set of notes.

Simple Moving Average

One of the most basic smoothing methods is the simple moving average. The simple moving average is computed by averaging a set number of observations with consecutive X-values in a particular window of neighborhood. It is common to use an odd number of points so that the calculation is symmetric. For example, to compute the smoothed Y-value (S_i) based on the 5-point simple moving average at X_i,

$$S_i = \frac{Y_{i-2} + Y_{i-1} + Y_i + Y_{i+1} + Y_{i+2}}{5}$$

which can be notated as

$$S_i = \frac{1}{5} \sum_{i-j}^{i+j} Y_i$$

where $j = {}^{k}-{}^{1}/_{2}$. To illustrate, we compute the 5-point simple moving average for Quarter 8 in the data.

$$\begin{aligned} S_8 &= \frac{1}{5} \sum_{6}^{10} Y_i \\ &= \frac{Y_6 + Y_7 + Y_8 + Y_9 + Y_{10}}{5} \\ &= \frac{3853289 + 3950326 + 3846413 + 4188955 + 2674918}{5} \\ &= 3702780 \end{aligned}$$

We can express the simple moving average as the dot product between a column vector of filtering weights ($\mathbf{w}$) and the observations in the neighborhood ($\mathbf{Y}$). For example, let

$$\mathbf{w} = \begin{bmatrix} 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} 3853289 \\ 3950326 \\ 3846413 \\ 4188955 \\ 2674918 \end{bmatrix}$$

The simplae moving average is then

$$S = \mathbf{w}^T \bullet \mathbf{Y}$$

Note that this type of smoothing works for scores in the middle of the distribution, but not for scores at the top of bottom (since one cannot go up past the first score nor down past the last one). For example,

$$\begin{aligned} S_2 &= \frac{Y_{-1} + Y_1 + Y_2 + Y_3 + Y_4}{5} \\ &= \frac{\text{NA} + 2962901 + 1875403 + 6569281 + 2780925}{5} \end{aligned}$$

It is unclear what to do with these endpoints. For example, if we sum the remaining cases and divide by five the smoothed value will probably be underestimated (too small). Another possibility is to divide by the number of neighboring points used (in this case 4), but that makes the windows/neighborhoods different for different X-values. Software typically omits these endpoints. Because our goal is to identify the overall trend, this is not problematic.

The simple moving average can be computed many ways using R. Here I use the `filter()` function from the **stats** package. Because I have the **dplyr** package loaded, I need to specify that the `filter()` function used is from the **stats** package and not that fro mthe **dplyr** package. The way we specify the number of points is via the `filter=` argument. Rather than giving the number of observations to use in the average, we give the coefficients used in the computation of the average for each of the observations; in our case each of the five coefficients is $1/5$.

```
# Add column of smoothed values
duff = duff %>%
  mutate(
    s = stats::filter(revenue, filter = c(1/5, 1/5, 1/5, 1/5, 1/5), side = 2)
        )

# Show data
duff
```

```
## # A tibble: 25 x 3
##    quarter  revenue        s
##      <int>    <dbl>    <dbl>
##  1       1 2962901.       NA
##  2       2 1875403.       NA
##  3       3 6569281. 3162110.
##  4       4 2780925. 3340188.
##  5       5 1622039. 3755172.
##  6       6 3853289. 3210599.
##  7       7 3950326. 3492205.
##  8       8 3846413. 3702780.
```

```
##  9       9 4188955. 4105771.
## 10      10 2674918. 4025658.
## # ... with 15 more rows
```

Below we plot the smoothed values versus their corresponding quarter. The smoothed values tend to show the overall trend more clearly than the observed data. This is because the "noise" is suppressed by the averaging. This smoothing also hides the cyclic nature of the data. Again, since we are interested in the overall trend, this is not problematic.

```
ggplot(data = duff, aes(x = quarter, y = revenue)) +
  geom_line(alpha = 0.7, color = "darkgray") +
  geom_line(aes(y = s), color = "red", size = 1.2) +
  theme_classic() +
  scale_x_continuous(name = "Quarter", breaks = seq(from = 0, to = 28, by = 4)) +
  ylab("Revenue")
```

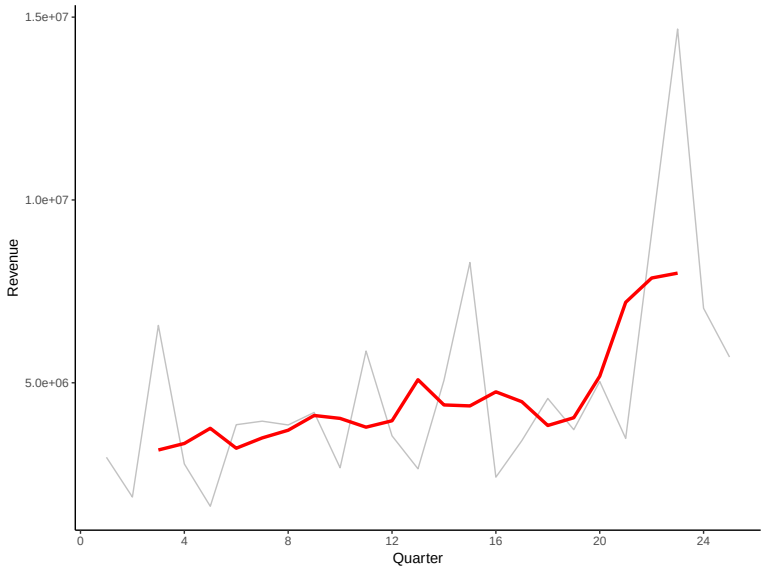

Figure 2: *Actual Revenue (dark-grey) and smoothed revenue using a 5-point simple moving average (red) by quarter for Duff Brewing Company.*

The choice of neighborhood affects the degree of smoothing. For example, the plot below shows a 3-point, 5-point, 7-point, and 9-point simple moving average. As we increase the number of observations in the neighborhood, the data are smoothed more and more. Often the choice of neighborhood is trial-and-error.

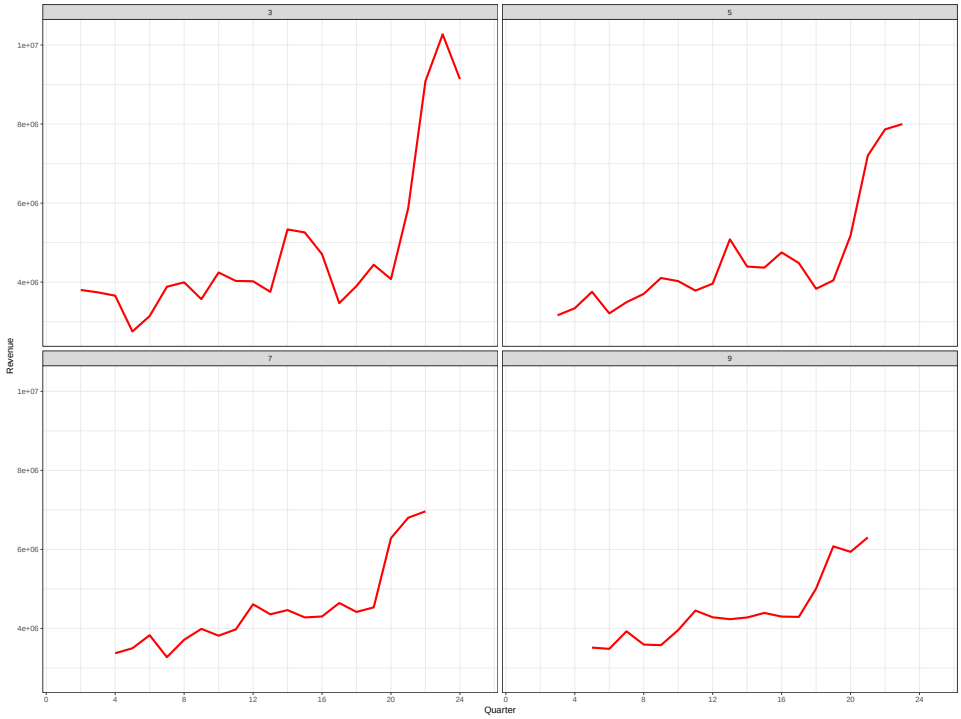

Figure 3: *Smoothed Revenue using a 3-point, 5-point, 7-point, and 9-point simple moving average by quarter for Duff Brewing Company.*

Weighted Moving Average

With the 5-point simple moving average, each observation in the neighborhood was given the same filtering weight, namely $1/5$. The weighted moving average allows different observations to have different weights. We typically give the value at X_i the highest weight, the next nearest neighbors get a smaller weight, and so forth. For example, with a five-point neighborhood, we could use the following

$$S_i = \frac{1(Y_{i-2}) + 2(Y_{i-1}) + 3(Y_i) + 2(Y_{i+1}) + 1(Y_{i+2})}{9}$$

Note since the average is now a weighted mean, we divide by the sum of the weights rather than the number of observations. The vector of filtering weights, $\mathbf{w}$, we use in the `fiter()` function are

$$\mathbf{w} = [\frac{1}{9}, \frac{2}{9}, \frac{1}{3}, \frac{2}{9}, \frac{1}{9}]$$

To compute the weighted means we again use the `filter()` function from the **stats** package.

```
# Add column of smoothed values
duff = duff %>%
  mutate(
    s_weight = stats::filter(revenue, filter = c(1/9, 2/9, 1/3, 2/9, 1/9), side = 2)
         )

# Show data
duff
```

```
## # A tibble: 25 x 4
##    quarter  revenue        s s_weight
##      <int>    <dbl>    <dbl>    <dbl>
##  1       1 2962901.       NA       NA
##  2       2 1875403.       NA       NA
##  3       3 6569281. 3162110. 3733938.
##  4       4 2780925. 3340188. 3383790.
##  5       5 1622039. 3755172. 3183795.
##  6       6 3853289. 3210599. 3259104.
##  7       7 3950326. 3492205. 3673486.
##  8       8 3846413. 3702780. 3816223.
##  9       9 4188955. 4105771. 3936455.
## 10      10 2674918. 4025658. 3948369.
## # ... with 15 more rows
```

Below we plot the smoothed values versus their corresponding quarter. The smoothed values tend to show the overall trend more clearly than the observed data. This is because the "noise" is suppressed by the averaging. This smoothing also hides the cyclic nature of the data. Again, since we are interested in the overall trend, this is not problematic.

```
ggplot(data = duff, aes(x = quarter, y = revenue)) +
  geom_line(alpha = 0.7, color = "darkgray") +
  geom_line(aes(y = s), color = "red", size = 1.2) +
  geom_line(aes(y = s_weight), color = "blue", size = 1.2) +
  theme_classic() +
  scale_x_continuous(name = "Quarter", breaks = seq(from = 0, to = 28, by = 4)) +
  ylab("Revenue")
```

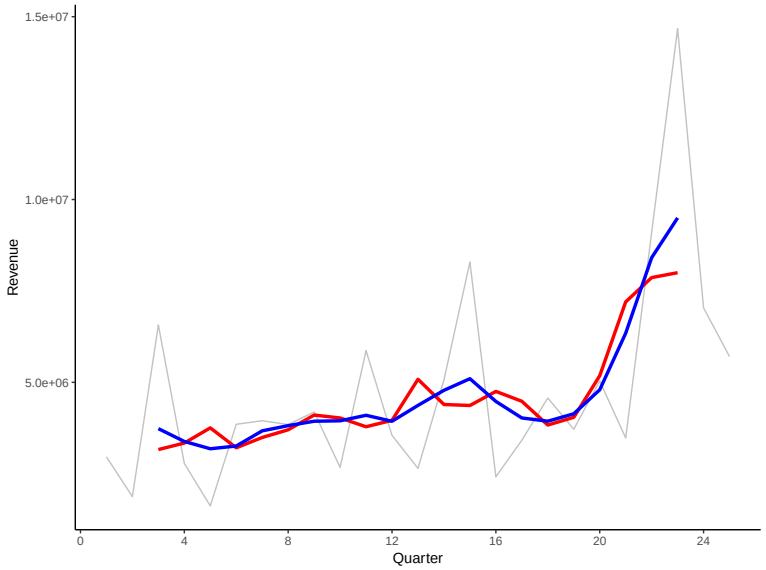

Figure 4: *Actual Revenue (dark-grey), smoothed revenue using a 5-point simple moving average (red), and weighted moving average (blue) by quarter for Duff Brewing Company.*

The weighting scheme we chose was linear around the observation (i.e., constant decrease in the weight). This is not the only weighting scheme we could have chosen. Other common weighting schemes include polynomial weighting, binomial weighting, and exponential weighting.

LOESS Smoothing

Locally estimated scatterplot smoothing (Loess), is another popular smoothing technique. Loess fits a series of simple models to localized subsets of the data to build up a smoother.

The localized subset of data is selected using a nearest neighbors algorithm of nq (rounded to the next largest integer) points where the value of q is the proportion of data used in each fit. (Note: The value of q, called the *bandwidth*, is selected by the analyst.) In our example, if we selected $q = .20$ then each localized set of data would be comprised of $25 * .20 = 5$ observations; the observation we are predicting for and the four nearest neighbors to that observation.

A low degree polynomial equation (often just linear) is then fitted to the localized data using weighted least squares. This resulting equation is then used to predict the value for the selected point. The process is repeated, moving to the next highest X-value again finding the 5 nearest observations, fitting a weighted regression, comoputing the prediction for this second point, and so on. The resulting points are then connected together with a line.

The weights used in the regression are typically computed as follows:

- Determine the distance from each point to the point of estimation;
- Scale the distances by the maximum distance over all points in the local data set;
- Compute the weights by evaluating the tricube weight function using the scaled distances;

where the tricube weight function is defined as

$$w(x) = \begin{cases} (1 - |x|^3)^3 & \text{for } |x| < 1 \\ 0 & \text{for } |x| \geq 1 \end{cases}$$

Let's look at our data. The first observation is (1, 2962901). If we use $q = .2$ then the 5 observations in the localized neighborhood to the first observation are:

```
## # A tibble: 5 x 2
##   quarter  revenue
##     <int>    <dbl>
## 1       1 2962901.
## 2       2 1875403.
## 3       3 6569281.
## 4       4 2780925.
## 5       5 1622039.
```

Since we are estimating the value at $X = 1$ (Quarter 1), we compute the distance to that X-value for each observation in the neighborhood.

```
## # A tibble: 5 x 3
##   quarter  revenue  dist
##     <int>    <dbl> <dbl>
## 1       1 2962901.     0
## 2       2 1875403.     1
## 3       3 6569281.     2
## 4       4 2780925.     3
## 5       5 1622039.     4
```

Next, we scale the distances by the maximum distance over all points in the local data set. Since the maximum distance is 4 we divide each distance by 4.

```
## # A tibble: 5 x 4
##   quarter  revenue  dist scl_dist
##     <int>    <dbl> <dbl>    <dbl>
## 1       1 2962901.     0     0
## 2       2 1875403.     1     0.25
## 3       3 6569281.     2     0.5
## 4       4 2780925.     3     0.75
## 5       5 1622039.     4     1
```

Now we can compute the weights we will use in our regression by evaluating the tricube weight function using the scaled distances. Any scaled distance that is less than 1 will be given a weight equal to $(1 - |\text{scl\_dist}|^3)^3$, and those with a scaled distance greater than or equal to 1 will get a weight of 0.

```
## # A tibble: 5 x 5
##   quarter  revenue  dist scl_dist     w
##     <int>    <dbl> <dbl>    <dbl> <dbl>
## 1       1 2962901.     0     0    1
## 2       2 1875403.     1     0.25 1.000
## 3       3 6569281.     2     0.5  0.998
## 4       4 2780925.     3     0.75 0.925
## 5       5 1622039.     4     1    0
```

We can now fit a linear model using weighted least squares predicting revenue from quarter in the localized neighborhood.

References