STRINGFEST ANALYTICS

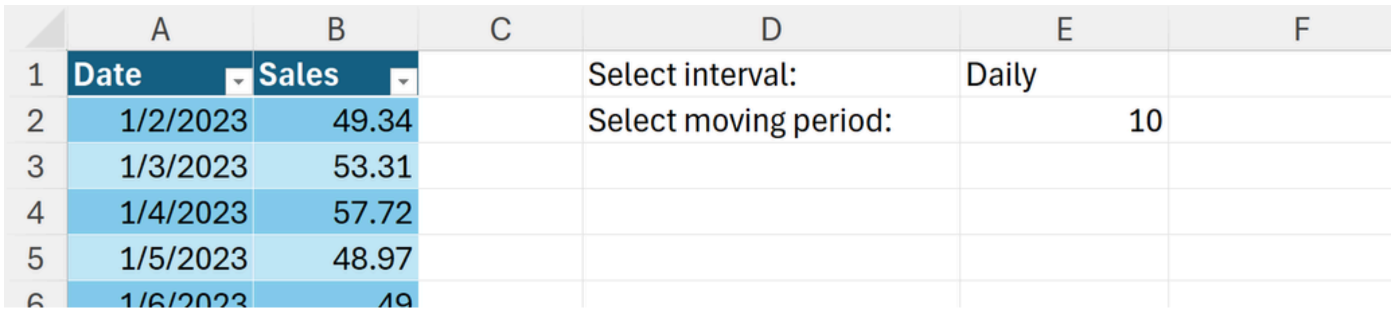
Python in Excel: How to create interactive visualizations

One of the greatest wins for Excel users learning Python is access to a comprehensive suite of data visualization tools. However, compared to Excel, the default plots produced with Python in Excel can seem somewhat static. For example, there's no straightforward way to add tooltips or interactive elements that allow users to click on parts of the chart to understand the data's origins.

Nonetheless, it is possible to make Python plots created in Excel feel more interactive and user-friendly. This post expands on a previous article where I explored using moving averages with Python in Excel. We'll extend this concept further to develop an interactive plot. You can follow along using the exercise file provided below:

DOWNLOAD THE EXERCISE FILE HERE

The first step I will take is to incorporate several user inputs that will guide the remainder of the model. Specifically, I will allow the user to determine the sampling interval for the data and the number of periods to use for the moving average:



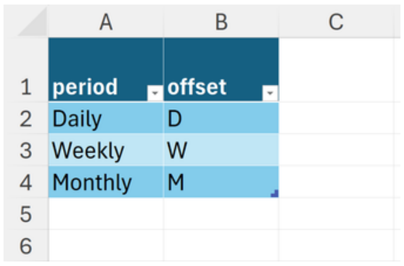
	A	B	C	D	E	F
1	Date	Sales		Select interval:	Daily	
2	1/2/2023	49.34		Select moving period:	10	
3	1/3/2023	53.31				
4	1/4/2023	57.72				
5	1/5/2023	48.97				
6	1/6/2023	49				

In an Excel model that requires user-defined inputs, it is a good idea to incorporate data validation controls to enhance usability and robustness. Specifically, I will include a dropdown menu to designate permissible interval periods within the model (“Daily,” “Monthly,” etc.). Accurate spelling of these intervals is essential, as it will dictate the subsequent operations of the model.

Therefore, I will create a lookup table that serves two purposes: it will identify the corresponding pandas date offset values required for Python integration, and establish a foundation of accuracy for the data validation. This approach ensures both precision and ease of use, streamlining the model's functionality.

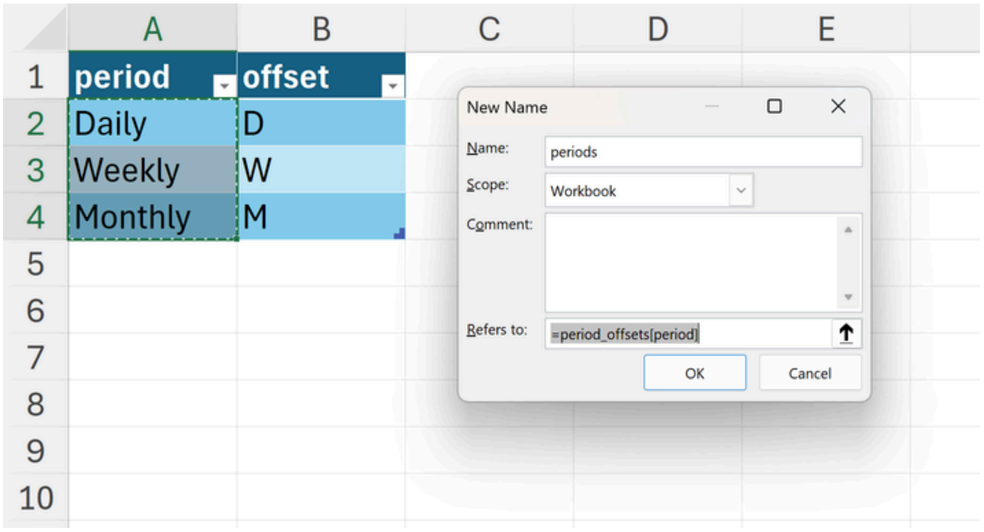
In a new worksheet, go ahead and list out the three options we are going to let the user choose between: Daily, Weekly and Monthly. Then in the next column, add the proper pandas date offset frequency string. In pandas, a date offset frequency string specifies the increment at which to apply a frequency conversion or generate a date range. These strings represent base time frequencies, such as "D" for day, "M" for month, and "W" for minute, with additional modifiers (like "S" for start or "E" for end) to indicate specific points within the given period.

Your table should look like this. Name it period_offsets:



	A	B	C
1	period	offset	
2	Daily	D	
3	Weekly	W	
4	Monthly	M	
5			
6			

Now that we've established our "source of truth," we'd like to use the period column as the basis for the data validation dropdown. This involves the slightly roundabout step of creating a named range from the Period table and using it as the foundation for the data validation dropdown. For more information, check out this post from MyOnlineTrainingHub.



	A	B
1	period	offset
2	Daily	D
3	Weekly	W
4	Monthly	M

New Name

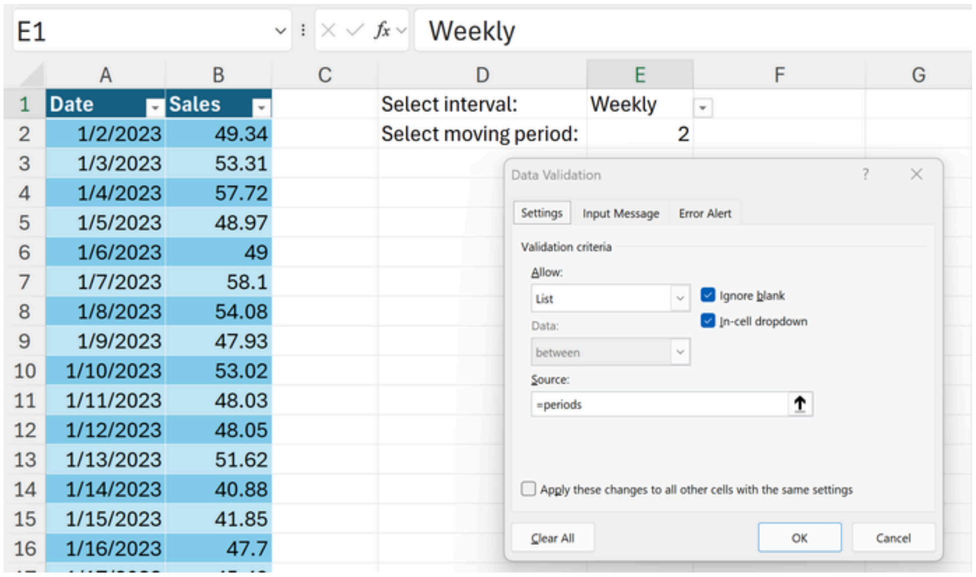
Name: periods

Scope: Workbook

Comment:

Refers to: =period_offsets[period]

After you have created the named range periods, select cell E2 in your main worksheet. Then, go to the ribbon, click on “Data,” and choose “Data Validation” to apply a dropdown list based on this range to your cell.



E1 | Weekly

	A	B	C	D	E
1	Date	Sales		Select interval:	Weekly
2	1/2/2023	49.34		Select moving period:	2
3	1/3/2023	53.31			
4	1/4/2023	57.72			
5	1/5/2023	48.97			
6	1/6/2023	49			
7	1/7/2023	58.1			
8	1/8/2023	54.08			
9	1/9/2023	47.93			
10	1/10/2023	53.02			
11	1/11/2023	48.03			
12	1/12/2023	48.05			
13	1/13/2023	51.62			
14	1/14/2023	40.88			
15	1/15/2023	41.85			
16	1/16/2023	47.7			

Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

[✓] Ignore blank

[✓] In-cell dropdown

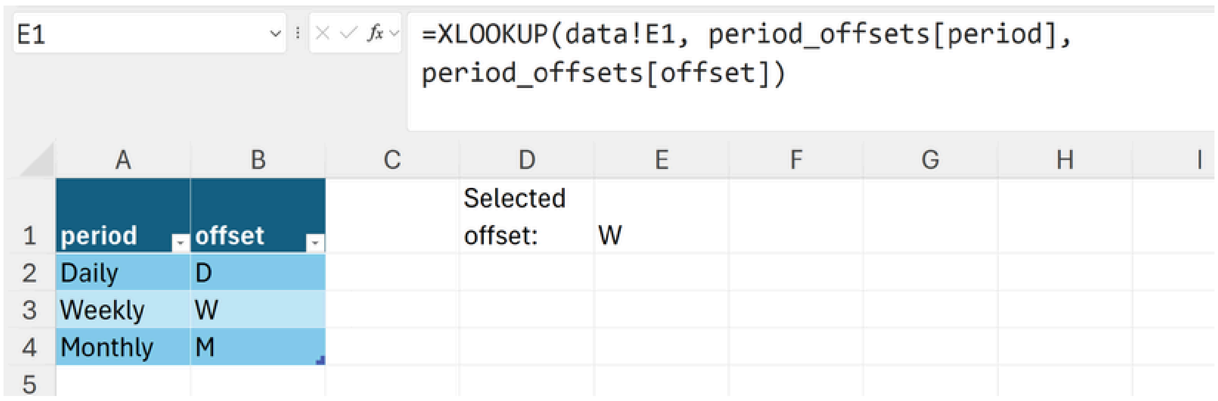
Data: between

Source: =periods

[ ] Apply these changes to all other cells with the same settings

Clear All | OK | Cancel

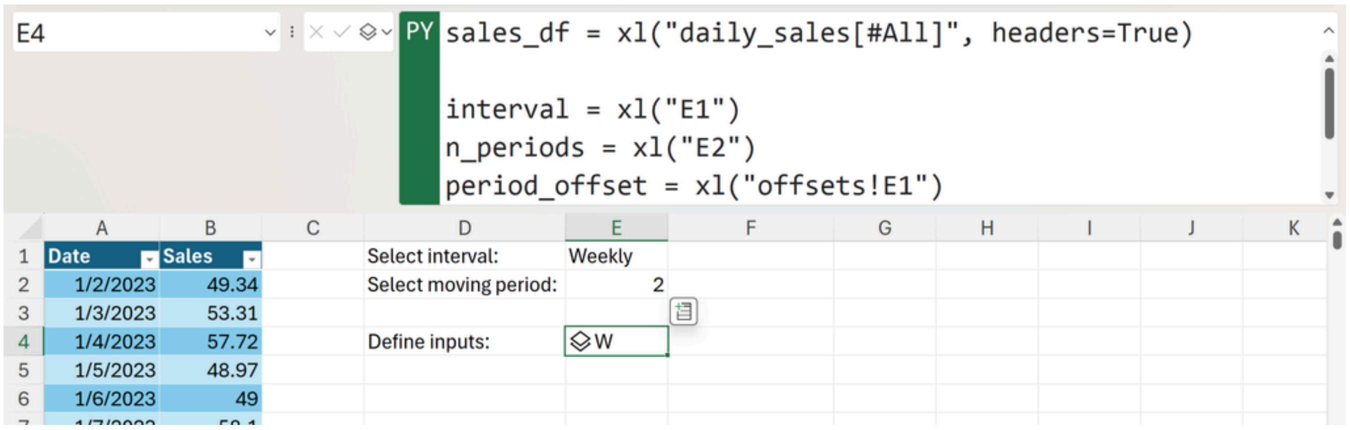
Excellent. We only need to define one more input, which will be offstage as it derives from the user’s selection. I plan to use the XLOOKUP() function to identify the selected date offset frequency string. This will then be passed into Python to dictate the resampling of the time series data.



```
=XLOOKUP(data!E1, period_offsets[period],
period_offsets[offset])
```

	A	B	C	D	E
1	period	offset		Selected offset:	W
2	Daily	D			
3	Weekly	W			
4	Monthly	M			

Great job! We now have all the data and user-defined inputs necessary to create this chart. Let's proceed by adding a Python cell that will load both the main dataset and the user-defined inputs as named objects:



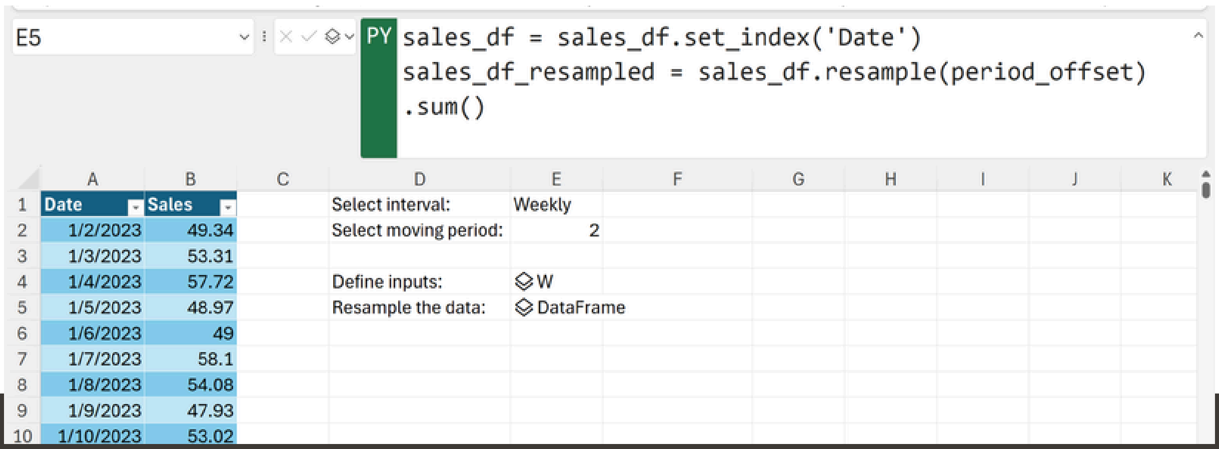
```
sales_df = xl("daily_sales[#All]", headers=True)

interval = xl("E1")
n_periods = xl("E2")
period_offset = xl("offsets!E1")
```

	A	B	C	D	E
1	Date	Sales		Select interval:	Weekly
2	1/2/2023	49.34		Select moving period:	2
3	1/3/2023	53.31			
4	1/4/2023	57.72		Define inputs:	W
5	1/5/2023	48.97			
6	1/6/2023	49			

Next, we will resample the dataset based on whichever interval the user requested.

First, resampling in pandas requires the date column to be set as the index of the DataFrame, so I'll do that using set_index().

Then, I will use the resample() method to calculate total sales over the specified interval. Remember, this is the value found in the period_offset variable, so we will pass that into resample() to obtain the desired resampling. I will then chain the sum() method to get the total sales for the specified period. If you want to further validate and experiment with this code, you can preview or display the resulting Excel values in the downloaded exercise file.



E5 PY

```
sales_df = sales_df.set_index('Date')
sales_df_resampled = sales_df.resample(period_offset)
.sum()
```

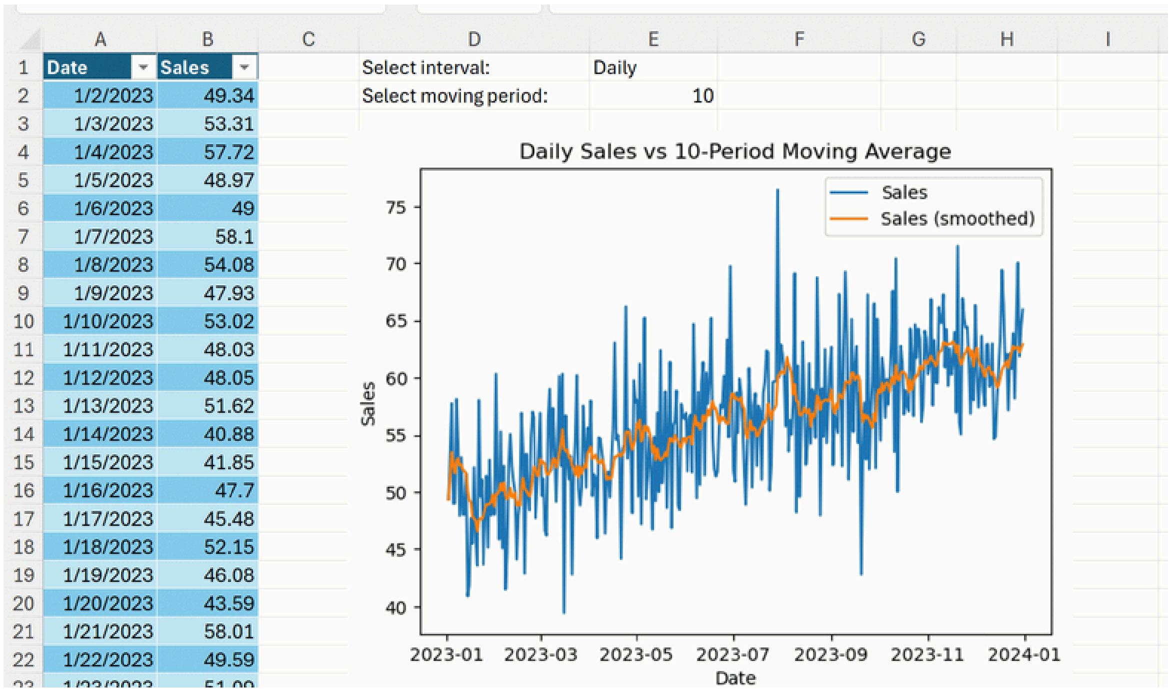
	A	B	C	D	E
1	Date	Sales		Select interval:	Weekly
2	1/2/2023	49.34		Select moving period:	2
3	1/3/2023	53.31			
4	1/4/2023	57.72		Define inputs:	W
5	1/5/2023	48.97		Resample the data:	DataFrame
6	1/6/2023	49			
7	1/7/2023	58.1			
8	1/8/2023	54.08			
9	1/9/2023	47.93			
10	1/10/2023	53.02			

The next step is to construct the plot, which only requires a few lines of code. I will plot the actual, “unsmoothed” resampled data, then add the smoothed sales data.

The trickier, more dynamic aspect here is the title of the plot. I would like it to dynamically be relabeled based on the period that is being resampled and the number of periods being used to smooth. I will use f-strings from Python to make this happen.

```
SNS.LINEPLOT(DATA=SALES_DF_RESAMPLED,
X='DATE', Y='SALES', LABEL='SALES')

SNS.LINEPLOT(DATA=SALES_DF_RESAMPLED,
X='DATE', Y='SALES SMOOTHED',
LABEL='SALES (SMOOTHED)')

PLT.TITLE(F"{INTERVAL} SALES VS
{N_PERIODS}-PERIOD MOVING AVERAGE")
```

Go ahead and display and resize the Python plot in Excel, and you should see something like the following. The user now controls both the sampling interval and the number of smoothing periods, and the plot updates accordingly.



Date	Sales
1/2/2023	49.34
1/3/2023	53.31
1/4/2023	57.72
1/5/2023	48.97
1/6/2023	49
1/7/2023	58.1
1/8/2023	54.08
1/9/2023	47.93
1/10/2023	53.02
1/11/2023	48.03
1/12/2023	48.05
1/13/2023	51.62
1/14/2023	40.88
1/15/2023	41.85
1/16/2023	47.7
1/17/2023	45.48
1/18/2023	52.15
1/19/2023	46.08
1/20/2023	43.59
1/21/2023	58.01
1/22/2023	49.59

Select interval:	Daily
Select moving period:	10

While the resulting plot still doesn’t have tooltips or the ability to click into the data to understand its series, the ability to work smoothly with time series data and easily customize the plot makes this a very worthwhile skill to have in your toolkit.

STRINGFEST ANALYTICS

THANK YOU

Thanks for checking out this post on building interactive data visualizations with Python in Excel!

Do you have any questions? Let me know in the comments. And if you're interested in getting started with Python as an Excel user, check out my book, *Advancing into Analytics: From Excel to Python and R.*