**Reproduce and fix a bug from an open-source data project’s GitHub issues**

**Pandas (data analysis library)**

pandas-dev/pandas is a widely used open-source Python library for data manipulation and analysis(45.4k★ on GitHub). It provides fast, flexible data structures (DataFrame/Series) and many statistical functions. An open issue (#61416) on this repo reports a bug in rolling window calculations (specifically for .std, .skew, and .kurt) when extreme values are present. This issue is reproducible and still open as of May 2025.

**Bug description**

The reporter found that excluding or including the first data point changes the end results of a rolling calculation – which should not happen for a fixed-window statistic. In their words, “the sliding window calculation shouldn’t be affected by data outside the window,” so including or excluding the first entry “should not affect the calculation result at the last index”. However, their example shows that the tail of the rolling kurtosis differs drastically between the two cases. In short, an outlier early in the series is improperly influencing later window results, which the pandas developers acknowledge “definitely shouldn’t happen”.

**Reproduction steps**

The issue is easily reproduced with a small example in Python. For example:

***import pandas as pd***

***# 1) Create a DataFrame of 100 values and introduce extreme outliers:***

***df = pd.DataFrame(index=range(100))***

***df['val'] = df.index***

***df = df/1e3***

***df.loc[0,'val'] = 1e6 # large positive outlier at index 0***

***df.loc[5,'val'] = -1e6 # large negative outlier at index 5***

***# 2) Compute rolling kurtosis with window size 20 (min\_periods=1):***

***res1 = df.rolling(20, min\_periods=1).kurt()***

***res2 = df.iloc[1:].rolling(20, min\_periods=1).kurt()***

***# 3) Compare the last few values:***

***print(res1.tail(3))***

***print(res2.tail(3))***

***With pandas 2.x, res1.tail() might show very large kurtosis values (e.g. ~722, 730, …) while res2.tail() shows constant -1.2***

The only difference is that res2 omits the first row. All other inputs (except the shifted window) are identical. In a correct implementation, the final few outputs should be the same, since neither window includes the extreme value at index 0. The fact that they differ confirms a bug.

**Observed vs Expected behavior**

* **Actual:** Including the first outlier (index 0) makes the tail values of .rolling(kurtosis) enormously different than if the first row is dropped. In the above example, one case ends in ~+750, the other in -1.2 (only the second is correct).
* **Expected:** For a fixed window, once the outlier falls out of the window, its effect should disappear. The reporter notes that “the sliding window calculation shouldn’t be affected by data outside the window”. In other words, adding or removing the first point (which is long before the last window) should not change the last window’s result.

**Cause (rolling algorithms)**

The pandas developers confirm that this is indeed a bug caused by the online (streaming) algorithm used for rolling std/skew/kurtosis. As one maintainer explains, “It looks like the huge outlier is influencing values outside of its window with .std, .skew, and .kurt”. In fact, a related issue (#60053) explicitly notes that “pandas uses online algorithms for some rolling functions like std, skew, etc. When your data has some extremas, these algorithms may yield inaccurate results.”. The pandas docs even warn: “large values may have an impact on windows which do not include these values” due to numerical imprecision. In short, the incremental calculation accumulators overflow or lose precision when faced with extreme outliers, so the results leak beyond their window.

**Workaround / Fix**

A practical workaround is to use a non-streaming calculation. For example, applying a custom aggregator (using NumPy or pandas functions) gives correct, consistent results. In this example, using .apply with NumPy’s functions yields identical outputs for res1 and res2 (both ending at –1.2), avoiding the bug (though at a performance cost):

***import numpy as np***

***# Correct approach using apply (slower):***

***res1\_fixed = df.rolling(20, min\_periods=1).apply(lambda x: pd.Series(x).kurt())***

***res2\_fixed = df.iloc[1:].rolling(20, min\_periods=1).apply(lambda x: pd.Series(x).kurt())***

The maintainers encourage a proper fix (e.g. a more numerically stable algorithm) in pandas’ rolling code. As of May 2025, issue #61416 remains open. Contributors can follow or comment on the discussion on GitHub:

* **Repository:** pandas-dev/pandas (stars: 45.4k)
* **Issue:** #61416 – “df.rolling.{std, skew, kurt} gives unexpected value”
* **Sources:** The issue report and discussion in the pandas repository.