How to create a union of pandas.Interval objects

Asked 2 years, 2 months ago Modified 9 months ago Viewed 650 times

1

Suppose I have the following list of ranges, all closed on the same side, stored as `pandas.Interval` objects

```
[[0, 5), [5, 10), [15, 20), [18, 24)]
```

Assuming that the list is already sorted (or is already coming from a `pandas.arrays.IntervalArray` object), how do I produce a `pandas.arrays.IntervalArray` whose constituent intervals are in the form of

```
[[0, 10), [15, 24)]
```

That is, the `repr` of the `IntervalArray` should print

```
<IntervalArray>
[[0, 10), [15, 24)]
Length: 2, closed: left, dtype: interval[int64]
```

instead of

```
<IntervalArray>
[[0, 5), [5, 10), [15, 20), [18, 24))]
Length: 4, closed: left, dtype: interval[int64]
```

Of course, I can manually compare each interval, and then create new `Interval` objects. Currently I am using [`more_itertools.split_when`](https://more-itertools.readthedocs.io/en/stable/api.html#more_itertools.split_when) :

```
intervals: List[Interval]

# Split `intervals` into groups whenever a pair of Intervals are disjoint
>>> map(IntervalArray, split_when(intervals, lambda x, y: not x.overlaps(y)))
```

The caveats of this method are:

1. This creates a list of `IntervalArray` s instead of a single one (and unfortunately you cannot create an `IntervalArray` directly from a list of `IntervalArray` s)*
2. This requires that all intervals be closed on both sides: it will split the ranges `[0, 5), [5, 0)` , even though they will be continuous when unioned.

I was wondering if there are ways to do this using `pandas` functions, such as `pandas.aggregate`.

I am aware that both versions of `IntervalArray` s will function exactly the same when used as indexes or for overlap checking. However, eventually I would like to persist the `IntervalArray` to a database using two columns of `INTEGER` s denoting the left and right side of the intervals, and an `IntervalArray` whose subintervals are unioned will produce much fewer pairs and requires much fewer rows.

*I mean if I really want to go down the one-liner path I can write:

```
IntervalArray([*map(lambda ia: Interval(ia.left[0], ia.right[-1], closed=ia.closed),
map(IntervalArray, split_when(intervals, lambda x, y: not x.overlaps(y))))])
```

But are there equivalent ways using tools in `pandas` ?

python pandas intervals

Share Improve this question Follow edited Jan 23, 2021 at 20:26 asked Jan 23, 2021 at 20:17

tonywu7
103 1 5

1 you may be interested in portion - it can create unions of intervals but you'll have to do the conversion between pandas and portion Intervals (back and forth) yourself – Stef Jan 23, 2021 at 22:26

@Stef `portion` is nice! I actually was solely using it, before I started moving everything to `pandas`. Its interval operations did allow a lot of possibilities – tonywu7 Jan 23, 2021 at 22:42

1 Answer

Sorted by: Highest score (default)

1

You can use a new python package called piso for this, which is designed for pandas. Install with pip or conda. Then using your example

```
import piso
piso.register_accessors()  # needed if using piso accessor

intervals = pd.arrays.IntervalArray.from_tuples(
    [(0, 5), (5, 10), (15, 20), (18, 24)],
    closed = "left",
)

result = intervals.piso.union()  # using accessor
```

```
# this works too
# result = piso.union(intervals)
```

the result is:

```
<IntervalArray>
[[0, 10), [15, 24)]
Length: 2, closed: left, dtype: interval[int64]
```

Piso will also work with other set operations like intersection, difference and others. It supports intervals with `pandas.Timestamp` and `pandas.Timedelta` values too.

note: I am the creator of piso. Please feel free to reach out with feedback or questions if you have any.

Share Improve this answer Follow edited Jun 21, 2022 at 3:50 answered Oct 11, 2021 at 5:44

Riley
2,093 1 5 14