Awkward-Pandas

Michael Hedges

April 17, 2019

Pandas

Great python tool for manipulating columns of NumPy arrays

- Tightly integrated with NumPy and SciPy
- Very popular
- Open source
- Large and active community

But...

Each column should be a standard NumPy type

- In otherwords, best use case is for flat tables/ntuples
- This means variable length arrays are prohibitively cumbersome at best
    - Usually not possible

Not HEP friendly

Pandas listens!

Pandas has developed an Extensions API

- Allows packages to extend pandas with custom arrays and datatypes
- No longer need to wait for pandas devs to support a data structure/type

Only requirement is that data type must be representable as a NumPy array

- AwkwardArray!

Extensions API

Arrays must be:

- Instances of pandas ExtensionArray
- Have unique type using pandas ExtensionDtype

```
>>> from pandas.api.extensions
import ExtensionArray, ExtensionDtype
```

If done properly, pandas will not coerce AwkwardArrays into NumPy object arrays (expensive!)

- Should then be able to store AwkwardArrays and access/manipulate them inside of pandas Dataframes

Awkward-Pandas

Awk-pandas branch of awkward-array on Github

- Very experimental
- Currently "forcing" AwkwardArray to use ExtensionArray as base class

Have successfully passed a JaggedArray to a DataFrame

- Can return JaggedArray
- Have created 'awkward' namespace in pandas dataframes objects

Testing Awkward-Pandas

Using example array from AwkwardArray README

```
array = awkward.fromiter([[1.1, 2.2, None, 3.3, None],
                          [4.4, [5.5]],
                          [{"x": 6, "y": {"z": 7}}, None, 
                         ])
df = pandas.DataFrame({'arr':array})
```

Testing Awkward-Pandas

Printing shows the dtype

```
>>> print(df)

0                              [1.1, 2.2, None, 3.3, None]
1                                                  [4.4, [5.5]]
2    [{'x': 6, 'y': {'z': 7}}, None, {'x': 8, 'y': ...
Name: arr, dtype: awkward
```

`df['arr'].values` **returns initial JaggedArray**

```
>>> print(df['arr'].values)
<JaggedArray [[1.1 2.2 None 3.3 None] [4.4 [5.5]] \
              [<Row 0> None <Row 1>]] at 0x0001218e8e48>
```

Testing Awkward-Pandas

Wrote custom accessor for AwkwardArray objects

```
df['arr'].awkward.__init__
<bound method AwkwardAccessor.__init__ of \
<awkward.pandas.accessor.AwkwardAccessor object \
at 0x121a70b38>>
```

Moving forward

Currently not very useful (just proof of concept)

- JaggedIndexing requires `df['arr'].values[:,:2]`
  - Doesn't work with `df['arr'][:,:2]`
- Need to figure out which AwkwardArray features need to be implemented
  - Decide what goes in accessor versus directly accessing `df['arr']`

Don't want to make pandas an explicit dependency of AwkwardArray

- This would affect uproot as well
- Could probably be done with `import awkward.pandas`
  - No pandas objects used unless explicitly imported