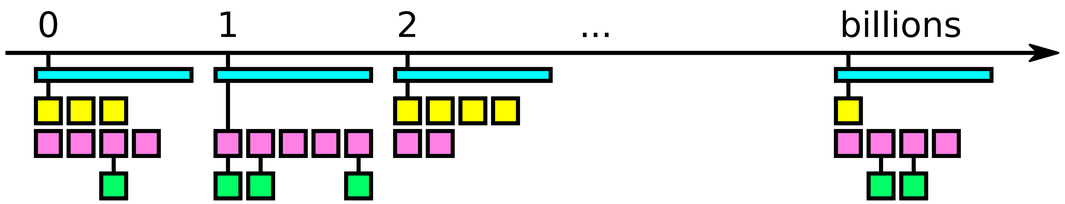
Awkward Array

An array library for **nested, variable-sized data**, including arbitrary-length lists, records, mixed types, and missing data, using **NumPy-like idioms**.



Arrays are **dynamically typed**, but operations on them are **compiled and fast**.

Example of an “awkward” array

```
array = ak.Array([
    [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],
    [],
    [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}]
])
```

Record structures with differently typed fields

Array of variable-length lists (“ragged” or “jagged“ arrays)

Nested variable-length lists

Missing data

Heterogenous data (union/variant types)

NumPy-like expression

```
output = np.square(array["y", ..., 1:])
```

Result

```
output.to_list()
   [
       [[], [4], [4, 9]],
       [],
       [[4, 9, 16], [4, 9, 16, 25]]
   ]
```

1.5 seconds
2.1 GB of memory

Equivalent Python

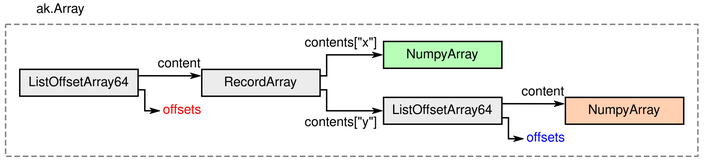
```
output = []
for sublist in python_objects:
    tmp1 = []
    for record in sublist:
        tmp2 = []
        for number in record["y"][1:]:
            tmp2.append(np.square(number))
        tmp1.append(tmp2)
    output.append(tmp1)
```

140 seconds
22 GB of memory

How it works

An entire array consists of **one small tree** with large, contiguous data buffers attached to each node (color coded with the above).

Compiled operations are performed on these data buffers, not the objects they represent.



(research scientists)


Jim Pivarski
Princeton University


Henry Schreiner
Princeton University

(research software engineers)


Ianna Osborne
Princeton University


Ioana Ifrim
Princeton University

(postdoc)


Angus Hollands
University of Birmingham
→ Princeton University

(undergraduates)


Anish Biswas
Manipal Institute of Technology


Manasvi Goyal
Delhi Technological University


Aryan Roy
Manipal Institute of Technology

(contractors)


Douglas Davis
Anaconda, Inc.


Martin Durant
Anaconda, Inc.


24 more contributors on GitHub