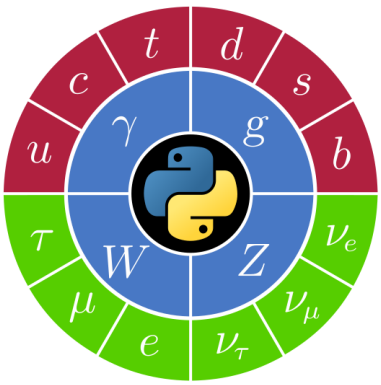
What's new with Vector?

Henry Schreiner (Princeton University)

Jim Pivarski (Princeton University)

Saransh Chopra (University of Delhi) □□□

PyHEP 2023



Quick recap!

Create and manipulate 2D, 3D, and Lorentz vectors -

```
pip install ve
conda install -c conda-forge
```

- Pure Python objects (numba-object)
- NumPy arrays of vectors
- Awkward Arrays of vectors (numpy-awkward)

Pure Python vectors

```
In [1]: v1 = vector.obj(x=3, y=4, z=-2, t=10)
In [2]: v2 = vector.obj(rho=5, phi=0.9273, eta=-0.39, t
In [3]: v1 + v2
Out[3]: VectorObject4D(x=5.999980871972148, y=8.0000143
In [4]: v1.x + v2.rho
Out[4]: 8
```

NumPy vectors

```
In [1]: v1 = vector.arr(
   ...:     {
   ...:         "x": [1.0, 2.0, 3.0, 4.0, 5
   ...:         "y": [1.1, 2.2, 3.3, 4.4, 5
   ...:         "z": [0.1, 0.2, 0.3, 0.4, 0
   ...:     }
   ...: )
In [2]: v2 = vector.arr(
   ...:     {
   ...:         "px": [1.0, 2.0, 3.0, 4.0,
   ...:         "py": [1.1, 2.2, 3.3, 4.4,
   ...:         "pz": [0.1, 0.2, 0.3, 0.4,
   ...:     }
   ...: )
In [3]: v1 + v2
Out[3]:
VectorNumpy3D([( 2.,  2.2, 0.2), ( 4.,  4.4
               ( 8.,  8.8, 0.8), (10., 11.,
              dtype=[('x', '<f8'), ('y', '<
In [4]: v1.x + v2.px
Out[4]: array([ 2.,  4.,  6.,  8., 10.])
```

```
In [1]: v1 = vector.awk(
   ...:     [
   ...:         [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2,
   ...:         [{"x": 3, "y": 3.3, "z": 0.3}],
   ...:         [{"x": 4, "y": 4.4, "z": 0.4}], {"x": 5,
   ...:     ]
   ...: )
In [2]: v2 = vector.awk(
   ...:     [
   ...:         [{"rho": 1, "phi": 1.1, "pz": 0.1}, {"rh
   ...:         [{"rho": 3, "phi": 3.3, "pz": 0.3}],
   ...:         [{"rho": 4, "phi": 4.4, "pz": 0.4}], {"rh
   ...:     ]
   ...: )
In [3]: v1 + v2
Out[3]: <VectorArray3D [[{x: 1.45, y: 1.99, ... z: 1}]]
In [4]: v1.x + v2.rho
Out[4]: <Array [[2, 4], [], [6], [8, 10]] type='4 * var
```

Awkward vectors

JIT compiled vectors

```
In [1]: @numba.njit
   ...: def compute_mass(v1,
   ...:     return (v1 + v2).

In [2]: compute_mass(
   ...:     vector.obj(px=1,
   ...:     vector.obj(px=-1,
   ...: )
Out[2]: 8.0
```

```
In [1]: array = vector.awk(
   ...:     [
   ...:         [
   ...:             dict(
   ...:                 {
   ...:                     x: numpy.random
   ...:                     for x in ("px",
   ...:                 },
   ...:                 E=numpy.random.norma
   ...:             )
   ...:             for inner in range(numpy
   ...:         ]
   ...:         for outer in range(50)
   ...:     ]
   ...: )

In [2]: @numba.njit
   ...: def compute_masses(array):
   ...:     out = numpy.empty(len(array), nu
   ...:     for i, event in enumerate(array
   ...:         total = vector.obj(px=0.0, p
   ...:         for vec in event:
   ...:             total = total + vec
   ...:         out[i] = total.mass
   ...:     return out
   ...:

In [3]: compute_masses(array)
Out[3]:
array([ 0.        , 10.36167953,  0.
        9.54993212,  0.        ,  9.0030842
```

Detailed resources

- PyHEP 2022 talk: Saransh-cpp/Constructing-HEP-vector-and- analyzing-HEP-data-using-Vector
- Vector's README: scikit-hep/vector
- Vector's Documentation: vector.readthedocs.io
- Vector's CHANGELOG: scikit-hep/vector/docs/changelog.md
- All talks - linked in README: README.md#talks-about-vector

But, what's new?

New constructors

```
In [1]: vector.VectorObject2D(x=2,
Out[1]: VectorObject2D(x=2, y=3)

In [2]: vector.MomentumObject2D(pt=
Out[2]: MomentumObject2D(pt=2, phi=

In [3]: vector.VectorObject3D.from_
Out[3]: VectorObject3D(x=2, y=3, z=

In [4]: vector.MomentumObject2D.fro
Out[4]: MomentumObject2D(pt=2, phi=

In [5]: from vector.backends.object
   ...: tZ, TemporalObjectT

In [6]: vector.VectorObject2D(azimu
   ...: ongitudinalObjectZ(4), temp
Out[6]: VectorObject4D(x=2, y=3, z=

In [7]: vector.MomentumObject4D(azi
   ...: =LongitudinalObjectZ(4), tem
Out[7]: MomentumObject4D(px=2, py=3
```

```
In [1]: vector.VectorNumpy2D([(1.1, 2.1), (1.2, 2.
   ...: 1.5, 2.5)], dtype=[('x', float), ('y', floa
Out[1]:
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3),
               dtype=[('x', '<f8'), ('y', '<f8')])

In [2]: vector.MomentumNumpy3D([(1.1, 2.1, 3.1), (
   ...: 3), (1.4, 2.4, 3.4), (1.5, 2.5, 3.5)], dtyp
   ...: at), ('pz', float)])
Out[2]:
MomentumNumpy3D([(1.1, 2.1, 3.1), (1.2, 2.2, 3.2),
                 (1.5, 2.5, 3.5)], dtype=[('x', '<
```

```
In [1]: vector.Array(
   ...:     [
   ...:         [{"x": 1, "y": 1.1, "z"
   ...:         {},
   ...:         [{"x": 3, "y": 3.3, "z"
   ...:         [
   ...:             {"x": 4, "y": 4.4,
   ...:             {"x": 5, "y": 5.5,
   ...:             {"x": 6, "y": 6.6,
   ...:         ],
   ...:     ]
   ...: )
Out[1]: <VectorArray3D [[{x: 1, y: 1.1

In [2]: vector.Array(
   ...:     [
   ...:         [{"px": 1, "py": 1.1, "
   ...:         {},
   ...:         [{"px": 3, "py": 3.3, "
   ...:         [
   ...:             {"px": 4, "py": 4.4
   ...:             {"px": 5, "py": 5.5
   ...:             {"px": 6, "py": 6.6
```

Awkward v2 support

Still using awkward v1? Still works with awkward v1!

No extra "vector" overhead for switching between awkward v1 to v2, everything is handled automatically

New features

User experience:	Better reprs, better error messages, better private-public API segregation, ...
Methods:	to_Vector*D, sum, count, count_nonzero, deltaRapidityPhi, deltaRapidityPhi2, ...
Support:	Python 3.11 and 3.12 support

Less annoying bugs

Checks: Type checks in constructors

Constructors: Bug fixes in old constructors

Properties and methods: numpy.sum, high coordinate values, ...

Revamped documentation

The documentation is now targeted towards users and not the developers of vector

A nice way to get involved!

- better layout for content
- more docstrings
- updated and tested examples
- doctests for public API
- better contributing guide
- and much more...

Where is vector now?

A stable place with v1.0.0 out

Development is being spearheaded by bug reports and feature requests, no specific development targets

Mostly voluntary development, participating in hacktoberfest!

Future plans (2024)

Possibility of making Vector differentiable using Jax

Ties up with making the AGC differentiable (AD)

```
@software{Schreiner_vector,
author = {Schreiner, Henry and Pivarski, Jim and Chopr
doi = {10.5281/zenodo.5942082},
license = {BSD-3-Clause},
title = {{vector}},
url = {https://github.com/scikit-hep/vector}
}
```

Thank you!

saransh-cpp.github.io

Speaker notes