Using Pandas with a Cartesian 3D Vector Class

https://github.com/t-o-k/scikit-vectors (https://github.com/t-o-k/scikit-vectors)

In [1]:

```
# This example has been tested with NumPy v1.15.3, Pandas v0.22.0 and Jupyter v4.4.0
```

In [2]:

```
from datetime import datetime
import numpy as np
import pandas as pd

from skvectors import create_class_Cartesian_3D_Vector
```

In [3]:

```
date_rng = pd.date_range(start='2017-01-01', end='2017-01-08', freq='H')

date_rng
```

Out[3]:

```
DatetimeIndex(['2017-01-01 00:00:00', '2017-01-01 01:00:00',
               '2017-01-01 02:00:00', '2017-01-01 03:00:00',
               '2017-01-01 04:00:00', '2017-01-01 05:00:00',
               '2017-01-01 06:00:00', '2017-01-01 07:00:00',
               '2017-01-01 08:00:00', '2017-01-01 09:00:00',
               ...
               '2017-01-07 15:00:00', '2017-01-07 16:00:00',
               '2017-01-07 17:00:00', '2017-01-07 18:00:00',
               '2017-01-07 19:00:00', '2017-01-07 20:00:00',
               '2017-01-07 21:00:00', '2017-01-07 22:00:00',
               '2017-01-07 23:00:00', '2017-01-08 00:00:00'],
              dtype='datetime64[ns]', length=169, freq='H')
```

In [4]:

```python
S3 = \
    create_class_Cartesian_3D_Vector(
        name = 'S3',
        component_names = 'xyz',
        brackets = '<>',
        sep = ', ',
        cnull = pd.Series(0, index=date_rng),
        cunit = pd.Series(1, index=date_rng),
        functions = \
            {
                'not': np.logical_not,
                'and': np.logical_and,
                'or': np.logical_or,
                'all': np.all,
                'any': np.any,
                'min': np.minimum,
                'max': np.maximum,
                'abs': np.absolute,
                'int': np.rint,
                'ceil': np.ceil,
                'copysign': np.copysign,
                'log10': np.log10,
                'cos': np.cos,
                'sin': np.sin,
                'atan2': np.arctan2,
                'pi': np.pi
            }
    )
```

In [5]:

```python
S3.component_null().head()
```

Out[5]:

```
2017-01-01 00:00:00    0
2017-01-01 01:00:00    0
2017-01-01 02:00:00    0
2017-01-01 03:00:00    0
2017-01-01 04:00:00    0
Freq: H, dtype: int64
```

In [6]:

```
S3.component_unit().head()
```

Out[6]:

```
2017-01-01 00:00:00    1
2017-01-01 01:00:00    1
2017-01-01 02:00:00    1
2017-01-01 03:00:00    1
2017-01-01 04:00:00    1
Freq: H, dtype: int64
```

In [7]:

```
clength = len(date_rng)

clength
```

Out[7]:

```
169
```

In [8]:

```
u = \
    S3(
        np.random.randint(0, 100, size=clength),
        np.random.randint(0, 100, size=clength),
        np.random.randint(0, 100, size=clength)
    )
u -= 50

u(pd.Series.head)
```

Out[8]:

```
S3(x=2017-01-01 00:00:00    31
2017-01-01 01:00:00   -41
2017-01-01 02:00:00    -4
2017-01-01 03:00:00    41
2017-01-01 04:00:00   -43
Freq: H, dtype: int64, y=2017-01-01 00:00:00    44
2017-01-01 01:00:00    41
2017-01-01 02:00:00    45
2017-01-01 03:00:00    25
2017-01-01 04:00:00   -44
Freq: H, dtype: int64, z=2017-01-01 00:00:00    -6
2017-01-01 01:00:00   -18
2017-01-01 02:00:00   -48
2017-01-01 03:00:00    34
2017-01-01 04:00:00    45
Freq: H, dtype: int64)
```

In [9]:

```
v = S3(1, 2, 3)

v(pd.Series.tail)
```

Out[9]:

```
S3(x=2017-01-07 20:00:00    1
2017-01-07 21:00:00    1
2017-01-07 22:00:00    1
2017-01-07 23:00:00    1
2017-01-08 00:00:00    1
Freq: H, dtype: int64, y=2017-01-07 20:00:00    2
2017-01-07 21:00:00    2
2017-01-07 22:00:00    2
2017-01-07 23:00:00    2
2017-01-08 00:00:00    2
Freq: H, dtype: int64, z=2017-01-07 20:00:00    3
2017-01-07 21:00:00    3
2017-01-07 22:00:00    3
2017-01-07 23:00:00    3
2017-01-08 00:00:00    3
Freq: H, dtype: int64)
```

In [10]:

```
w = u.cross(v).normalize()

w(pd.Series.tail)
```

Out[10]:

```
S3(x=2017-01-07 20:00:00   -0.374649
2017-01-07 21:00:00    0.793174
2017-01-07 22:00:00    0.756430
2017-01-07 23:00:00    0.880247
2017-01-08 00:00:00    0.142128
Freq: H, dtype: float64, y=2017-01-07 20:00:00    0.824228
2017-01-07 21:00:00    0.350472
2017-01-07 22:00:00   -0.631842
2017-01-07 23:00:00    0.203134
2017-01-08 00:00:00    0.801084
Freq: H, dtype: float64, z=2017-01-07 20:00:00   -0.424602
2017-01-07 21:00:00   -0.498040
2017-01-07 22:00:00    0.169084
2017-01-07 23:00:00   -0.428838
2017-01-08 00:00:00   -0.581432
Freq: H, dtype: float64)
```

In [11]:
```
c = 2.5 * w(np.ceil)

c(pd.Series.tail)
```

Out[11]:
```
S3(x=2017-01-07 20:00:00   -0.0
2017-01-07 21:00:00    2.5
2017-01-07 22:00:00    2.5
2017-01-07 23:00:00    2.5
2017-01-08 00:00:00    2.5
Freq: H, dtype: float64, y=2017-01-07 20:00:00    2.5
2017-01-07 21:00:00    2.5
2017-01-07 22:00:00   -0.0
2017-01-07 23:00:00    2.5
2017-01-08 00:00:00    2.5
Freq: H, dtype: float64, z=2017-01-07 20:00:00   -0.0
2017-01-07 21:00:00   -0.0
2017-01-07 22:00:00    2.5
2017-01-07 23:00:00   -0.0
2017-01-08 00:00:00   -0.0
Freq: H, dtype: float64)
```

In [12]:
```
w.x.tail()
```

Out[12]:
```
2017-01-07 20:00:00   -0.374649
2017-01-07 21:00:00    0.793174
2017-01-07 22:00:00    0.756430
2017-01-07 23:00:00    0.880247
2017-01-08 00:00:00    0.142128
Freq: H, dtype: float64
```

In [13]:
```
type(w.x)
```

Out[13]:
```
pandas.core.series.Series
```

In [14]:
```
w.x.index[-5:]
```

Out[14]:
```
DatetimeIndex(['2017-01-07 20:00:00', '2017-01-07 21:00:00',
               '2017-01-07 22:00:00', '2017-01-07 23:00:00',
               '2017-01-08 00:00:00'],
              dtype='datetime64[ns]', freq='H')
```

In [15]:

```
w.x.values[-5:]
```

Out[15]:

```
array([-0.37464893,  0.79317435,  0.75643016,  0.8802468 ,  0.14212788])
```

In [16]:

```
type(w.x.values)
```

Out[16]:

```
numpy.ndarray
```

In [17]:

```
df = pd.DataFrame(w.as_dict())

df.tail()
```

Out[17]:

	x	y	z
2017-01-07 20:00:00	-0.374649	0.824228	-0.424602
2017-01-07 21:00:00	0.793174	0.350472	-0.498040
2017-01-07 22:00:00	0.756430	-0.631842	0.169084
2017-01-07 23:00:00	0.880247	0.203134	-0.428838
2017-01-08 00:00:00	0.142128	0.801084	-0.581432

In [18]:

```
a = np.array(w).T

a[-5:]
```

Out[18]:

```
array([[-0.37464893,  0.82422765, -0.42460212],
       [ 0.79317435,  0.35047239, -0.49803971],
       [ 0.75643016, -0.63184166,  0.16908439],
       [ 0.8802468 ,  0.20313388, -0.42883819],
       [ 0.14212788,  0.80108441, -0.58143223]])
```

In [ ]:

```

```