Pynapple : Python Neural Analysis package

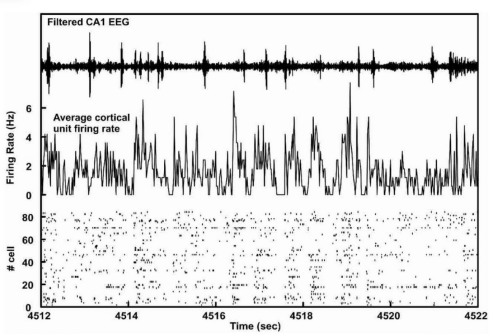
SFN Workshop, November 2025

Guillaume Viejo

FLATIRON INSTITUTE

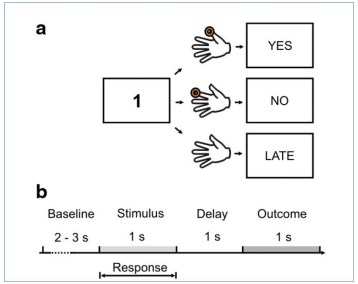
Where does pynapple comes from?

A possible classification of experiments



Hippocampal sharp wave bursts coincide with neocortical "up-state" transitions

Francesco P. Battaglia,[1] Gary R. Sutherland, and Bruce L. McNaughton[2]
Arizona Research Laboratories–Division of Neural Systems, Memory, and Aging, University of Arizona, Tucson, Arizona 85724, USA

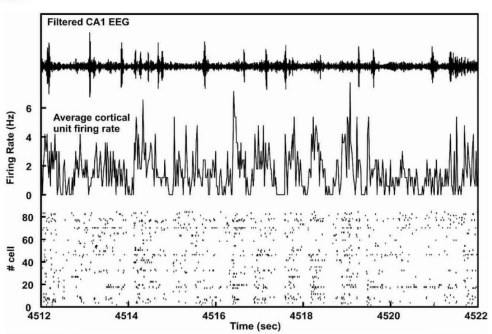


Behavioral/Cognitive

Characterization of Cortical Networks and Corticocortical Functional Connectivity Mediating Arbitrary Visuomotor Mapping

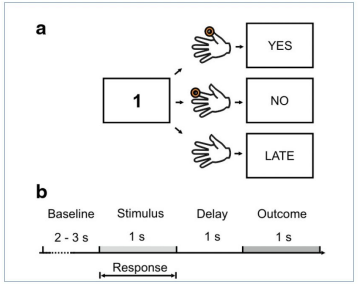
Andrea Brovelli,[1] Daniel Chicharro,[1] Jean-Michel Badier,[1,2] Huifang Wang,[1,3] and Viktor Jirsa[1,3]

A possible classification of experiments



Hippocampal sharp wave bursts coincide with neocortical "up-state" transitions

Francesco P. Battaglia,[1] Gary R. Sutherland, and Bruce L. McNaughton[2]
Arizona Research Laboratories–Division of Neural Systems, Memory, and Aging, University of Arizona, Tucson, Arizona 85724, USA



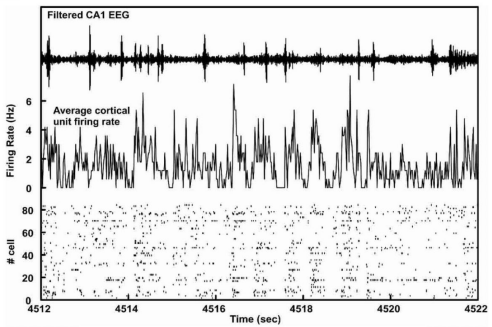
Behavioral/Cognitive

Characterization of Cortical Networks and Corticocortical Functional Connectivity Mediating Arbitrary Visuomotor Mapping

Andrea Brovelli,[1] Daniel Chicharro,[1] Jean-Michel Badier,[1,2] Huifang Wang,[1,3] and Viktor Jirsa[1,3]

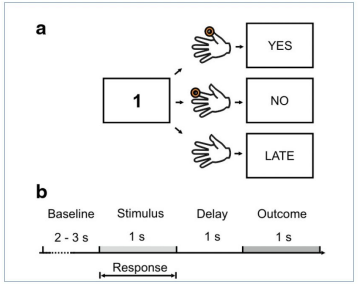
Less structured ⟷ More structured

A possible classification of experiments



Hippocampal sharp wave bursts coincide with neocortical "up-state" transitions

Francesco P. Battaglia,[1] Gary R. Sutherland, and Bruce L. McNaughton[2]
Arizona Research Laboratories–Division of Neural Systems, Memory, and Aging, University of Arizona, Tucson, Arizona 85724, USA



Behavioral/Cognitive

Characterization of Cortical Networks and Corticocortical Functional Connectivity Mediating Arbitrary Visuomotor Mapping

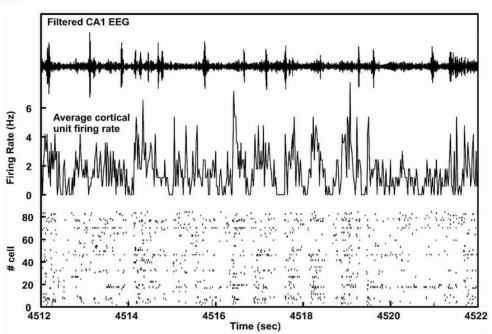
Andrea Brovelli,[1] Daniel Chicharro,[1] Jean-Michel Badier,[1,2] Huifang Wang,[1,3] and Viktor Jirsa[1,3]

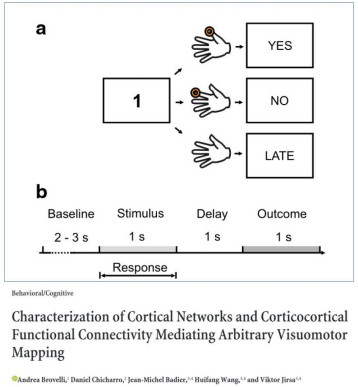
Less structured ⟷ More structured

→ TsToolbox (matlab)

Francesco Battaglia

A possible classification of experiments





Less structured ⟷ More structured

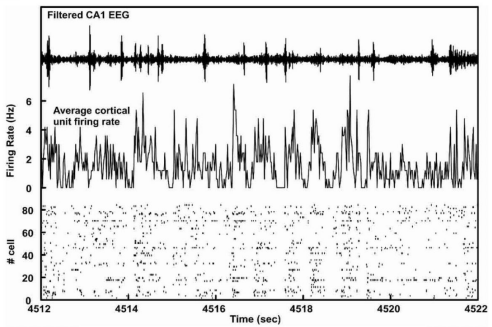
Francesco Battaglia

→ TsToolbox (matlab)

→ TsToolbox2

→ Neuroseries

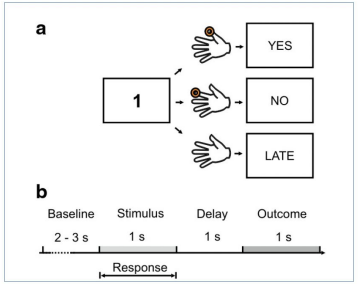
A possible classification of experiments



Hippocampal sharp wave bursts coincide with neocortical "up-state" transitions

Francesco P. Battaglia,[1] Gary R. Sutherland, and Bruce L. McNaughton[2]

Arizona Research Laboratories–Division of Neural Systems, Memory, and Aging, University of Arizona, Tucson, Arizona 85724, USA



Behavioral/Cognitive

Characterization of Cortical Networks and Corticocortical Functional Connectivity Mediating Arbitrary Visuomotor Mapping

Andrea Brovelli,[1] Daniel Chicharro,[1] Jean-Michel Badier,[1,2] Huifang Wang,[1,3] and Viktor Jirsa[1,3]

Less structured ⟷ More structured

Francesco Battaglia

→ TsToolbox (matlab)

→ TsToolbox2

→ Neuroseries → Pynapple

When do I need pynapple?

Time

Preprocessing
(CaImAn, SpikeInterface, …)

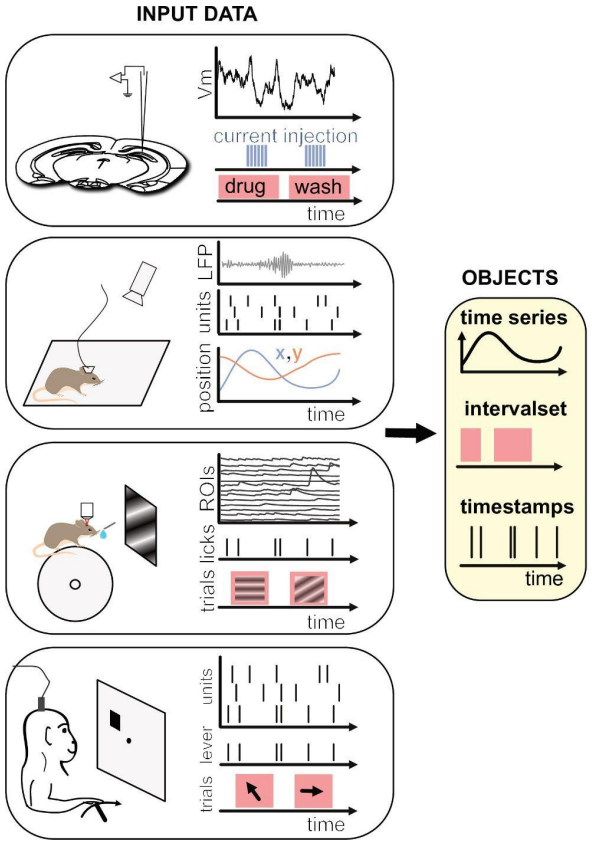
Postprocessing
(GLM, Manifold, …)

Time

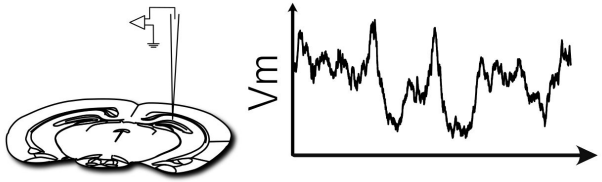
Preprocessing
(CaImAn, SpikeInterface, …)

Pynapple

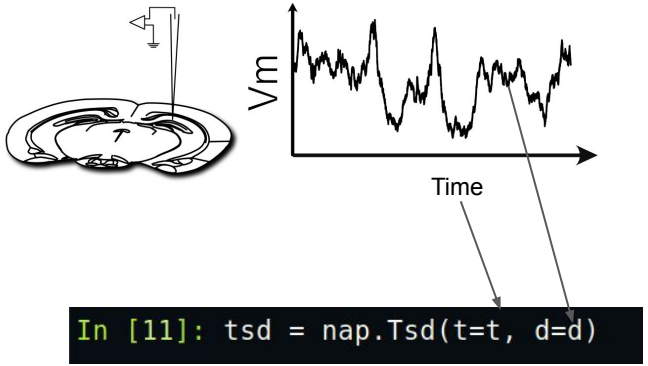
Postprocessing
(GLM, Manifold, …)

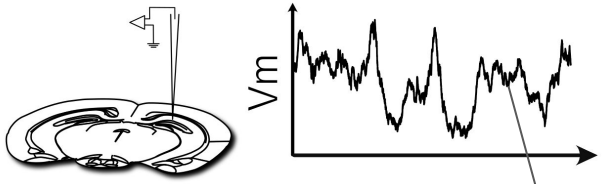


Time Series Data : Tsd, TsdFrame and TsdTensor

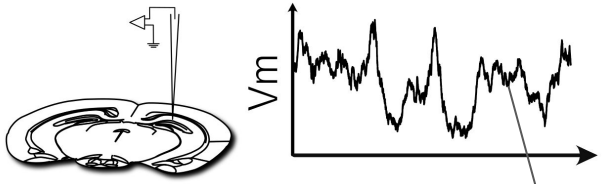


FLATIRON INSTITUTE


Vm
Time
In [11]: tsd = nap.Tsd(t=t, d=d)



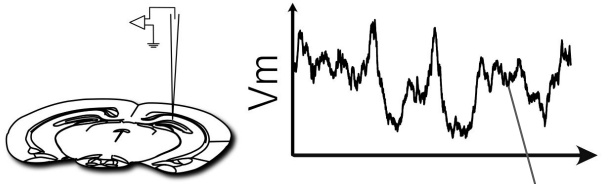

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```



Numpy array

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

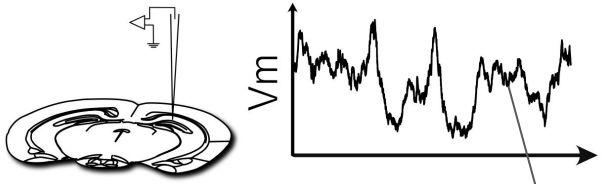


Numpy array

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```



Numpy array

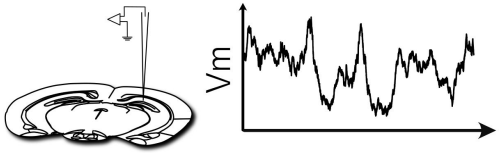
```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

You gain

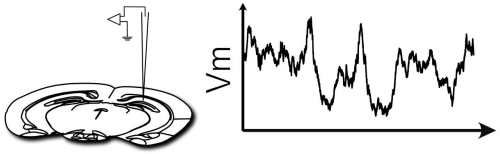
- Automatic handling of time units
- Epoch restriction
- Binning
- Time support
- …

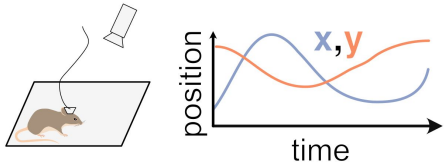


FLATIRON INSTITUTE

Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```



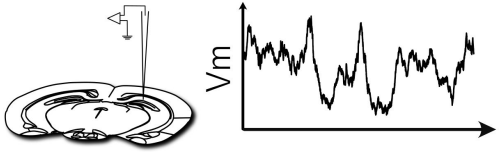


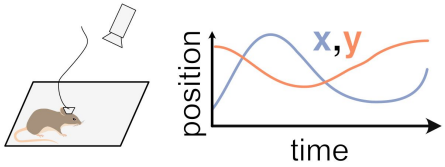
Tsd: 1-dimension

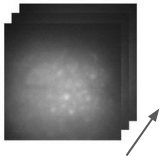
```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:     columns = ['x', 'y'])

In [21]: tsdframe
Out[21]:
Time (s)            x          y
----------  ---------  ---------
0.0         -0.029719  -0.273102
1.0          0.181754   3.25403
2.0         -0.495068  -0.524877
3.0         -1.20696   -0.033936
4.0         -0.664662   2.20862
...
95.0         0.942969   0.180585
96.0         2.15161    0.661736
97.0         0.751956  -1.72922
98.0        -1.45054    1.52954
99.0         0.199145   0.582944
dtype: float64, shape: (100, 2)
```







FLATIRON INSTITUTE

Tsd: 1-dimension

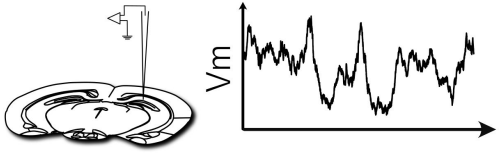
```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

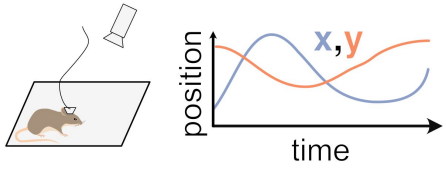
TsdFrame: 2-dimensions

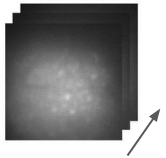
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])

In [21]: tsdframe
Out[21]:
Time (s)            x          y
----------  ---------  ---------
0.0         -0.029719  -0.273102
1.0          0.181754   3.25403
2.0         -0.495068  -0.524877
3.0         -1.20696   -0.033936
4.0         -0.664662   2.20862
...
95.0         0.942969   0.180585
96.0         2.15161    0.661736
97.0         0.751956  -1.72922
98.0        -1.45054    1.52954
99.0         0.199145   0.582944
dtype: float64, shape: (100, 2)
```

TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)

In [35]: tsdtensor
Out[35]:
Time (s)
----------  ----------------------
0.0         [[-0.87 ... -0.87] ...]
1.0         [[1.14 ... 1.14] ...]
2.0         [[0.25 ... 0.25] ...]
3.0         [[1.29 ... 1.29] ...]
4.0         [[-0.91 ... -0.91] ...]
...
95.0        [[-2.01 ... -2.01] ...]
96.0        [[-0.08 ... -0.08] ...]
97.0        [[0.53 ... 0.53] ...]
98.0        [[-0.62 ... -0.62] ...]
99.0        [[-0.55 ... -0.55] ...]
dtype: float64, shape: (100, 15, 15)
```







FLATIRON INSTITUTE

Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])

In [21]: tsdframe
Out[21]:
Time (s)            x          y
----------  ---------  ---------
0.0         -0.029719  -0.273102
1.0          0.181754   3.25403
2.0         -0.495068  -0.524877
3.0         -1.20696   -0.033936
4.0         -0.664662   2.20862
...
95.0         0.942969   0.180585
96.0         2.15161    0.661736
97.0         0.751956  -1.72922
98.0        -1.45054    1.52954
99.0         0.199145   0.582944
dtype: float64, shape: (100, 2)
```

TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)

In [35]: tsdtensor
Out[35]:
Time (s)
----------  ----------------------
0.0         [[-0.87 ... -0.87] ...]
1.0         [[1.14 ... 1.14] ...]
2.0         [[0.25 ... 0.25] ...]
3.0         [[1.29 ... 1.29] ...]
4.0         [[-0.91 ... -0.91] ...]
...
95.0        [[-2.01 ... -2.01] ...]
96.0        [[-0.08 ... -0.08] ...]
97.0        [[0.53 ... 0.53] ...]
98.0        [[-0.62 ... -0.62] ...]
99.0        [[-0.55 ... -0.55] ...]
dtype: float64, shape: (100, 15, 15)
```

```
In [41]: tsd.as_series()
```

```
In [42]: tsdframe.as_dataframe()
```

pandas

FLATIRON INSTITUTE

Doing math: the numpy array container approach

Numpy array

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```

FLATIRON INSTITUTE

Numpy array

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0          0.397043
1.0          1.55294
2.0          0.455892
3.0         -1.17359
4.0         -0.110113
...
95.0        -0.573408
96.0        -0.0110915
97.0        -1.58027
98.0         0.998846
99.0         0.542692
dtype: float64, shape: (100,)
```

Numpy functions

np.mean

np.add

np.min

...

Arithmetical operations

```
In [4]: tsd
Out[4]:
Time (s)
----------  --
0            0
1            1
2            2
3            3
4            4
dtype: int64, shape: (5,)
```

Arithmetical operations

```
In [4]: tsd
Out[4]:
Time (s)
----------  --
0            0
1            1
2            2
3            3
4            4
dtype: int64, shape: (5,)
```

```
In [5]: tsd + 1
Out[5]:
Time (s)
----------  --
0            1
1            2
2            3
3            4
4            5
dtype: int64, shape: (5,)
```

Arithmetical operations

```
In [4]: tsd
Out[4]:
Time (s)
----------  --
0            0
1            1
2            2
3            3
4            4
dtype: int64, shape: (5,)
```

```
In [5]: tsd + 1
Out[5]:
Time (s)
----------  --
0            1
1            2
2            3
3            4
4            5
dtype: int64, shape: (5,)
```

```
In [6]: tsd + np.array([0, 1, 2, 3, 4])
Out[6]:
Time (s)
----------  --
0            0
1            2
2            4
3            6
4            8
dtype: int64, shape: (5,)
```

Arithmetical operations

```
In [4]: tsd
Out[4]:
Time (s)
----------  --
0            0
1            1
2            2
3            3
4            4
dtype: int64, shape: (5,)
```

```
In [5]: tsd + 1
Out[5]:
Time (s)
----------  --
0            1
1            2
2            3
3            4
4            5
dtype: int64, shape: (5,)
```

```
In [6]: tsd + np.array([0, 1, 2, 3, 4])
Out[6]:
Time (s)
----------  --
0            0
1            2
2            4
3            6
4            8
dtype: int64, shape: (5,)
```

```
In [7]: tsd + tsd
---------------------------------
TypeError
Cell In[7], line 1
----> 1 tsd + tsd

File ~/miniconda3/envs/pynapple/li
lib/mixins.py:21, in _binary_metho
     19 if _disables_array_ufunc(o
     20     return NotImplemented
---> 21 return ufunc(self, other)
```

Array operations

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

FLATIRON INSTITUTE

Array operations

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

numpy.ndarray

```
In [18]: np.mean(tsdtensor, axis=0)
Out[18]:
array([[0.578, 0.468, 0.506],
       [0.508, 0.554, 0.352],
       [0.478, 0.274, 0.282],
       [0.206, 0.608, 0.426]])
```

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

numpy.ndarray

```
In [18]: np.mean(tsdtensor, axis=0)
Out[18]:
array([[0.578, 0.468, 0.506],
       [0.508, 0.554, 0.352],
       [0.478, 0.274, 0.282],
       [0.206, 0.608, 0.426]])
```

nap.TsdFrame

```
In [19]: np.mean(tsdtensor, axis=1)
Out[19]:
  Time (s)       0       1       2
----------  ------  ------  ------
         0  0.45    0.3325  0.6275
         1  0.5275  0.4875  0.2
         2  0.475   0.7225  0.315
         3  0.2875  0.4325  0.345
         4  0.4725  0.405   0.47
dtype: float64, shape: (5, 3)
```

Array slicing

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

FLATIRON INSTITUTE

Array slicing

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

numpy.ndarray

```
In [21]: tsdtensor[0]
Out[21]:
array([[0.65, 0.89, 0.94],
       [0.28, 0.25, 0.03],
       [0.26, 0.1 , 0.58],
       [0.61, 0.09, 0.96]])
```

Array slicing

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

numpy.ndarray

```
In [21]: tsdtensor[0]
Out[21]:
array([[0.65, 0.89, 0.94],
       [0.28, 0.25, 0.03],
       [0.26, 0.1 , 0.58],
       [0.61, 0.09, 0.96]])
```

nap.TsdTensor

```
In [22]: tsdtensor[0:2]
Out[22]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
dtype: float64, shape: (2, 4, 3)
```

Array slicing

```
In [17]: tsdtensor
Out[17]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
2           [[0.6 ... 0.6] ...]
3           [[0.54 ... 0.54] ...]
4           [[0.97 ... 0.97] ...]
dtype: float64, shape: (5, 4, 3)
```

numpy.ndarray

```
In [21]: tsdtensor[0]
Out[21]:
array([[0.65, 0.89, 0.94],
       [0.28, 0.25, 0.03],
       [0.26, 0.1 , 0.58],
       [0.61, 0.09, 0.96]])
```

nap.TsdTensor

```
In [22]: tsdtensor[0:2]
Out[22]:
Time (s)
----------  --------------------
0           [[0.65 ... 0.65] ...]
1           [[0.13 ... 0.13] ...]
dtype: float64, shape: (2, 4, 3)
```

nap.Tsd

```
In [24]: tsdtensor[:,0,0]
Out[24]:
Time (s)
----------  ----
0           0.65
1           0.13
2           0.6
3           0.54
4           0.97
dtype: float64, shape: (5,)
```

Array concatenation

```
In [15]: tsd1
Out[15]:
Time (s)
----------  ----------
0            0.910503
1           -0.0110368
2            0.496159
3           -0.956014
4           -0.592748
5           -0.711171
6           -0.370642
7            0.424684
8            0.718995
9            0.616419
dtype: float64, shape: (10,)
```

+

```
In [16]: tsd2
Out[16]:
Time (s)
----------  ---------
10           0.159056
11           1.10691
12           0.856208
13           1.44663
14          -2.11429
15          -1.01082
16           0.563591
17           2.09225
18           0.484394
19           0.482061
dtype: float64, shape: (10,)
```

Array concatenation

```
In [15]: tsd1
Out[15]:
Time (s)
----------  ----------
0            0.910503
1           -0.0110368
2            0.496159
3           -0.956014
4           -0.592748
5           -0.711171
6           -0.370642
7            0.424684
8            0.718995
9            0.616419
dtype: float64, shape: (10,)
```

+

```
In [16]: tsd2
Out[16]:
Time (s)
----------  ---------
10           0.159056
11           1.10691
12           0.856208
13           1.44663
14          -2.11429
15          -1.01082
16           0.563591
17           2.09225
18           0.484394
19           0.482061
dtype: float64, shape: (10,)
```

```
In [17]: np.concatenate((tsd1, tsd2))
Out[17]:
Time (s)
----------  ----------
0            0.910503
1           -0.0110368
2            0.496159
3           -0.956014
4           -0.592748
5           -0.711171
6           -0.370642
7            0.424684
8            0.718995
9            0.616419
10           0.159056
11           1.10691
12           0.856208
13           1.44663
14          -2.11429
15          -1.01082
16           0.563591
17           2.09225
18           0.484394
19           0.482061
dtype: float64, shape: (20,)
```

Array concatenation

```
In [15]: tsd1
Out[15]:
Time (s)
----------  ----------
0            0.910503
1           -0.0110368
2            0.496159
3           -0.956014
4           -0.592748
5           -0.711171
6           -0.370642
7            0.424684
8            0.718995
9            0.616419
dtype: float64, shape: (10,)
```

+

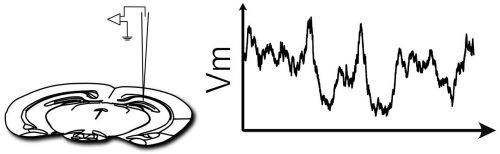
```
In [16]: tsd2
Out[16]:
Time (s)
----------  ---------
10           0.159056
11           1.10691
12           0.856208
13           1.44663
14          -2.11429
15          -1.01082
16           0.563591
17           2.09225
18           0.484394
19           0.482061
dtype: float64, shape: (10,)
```

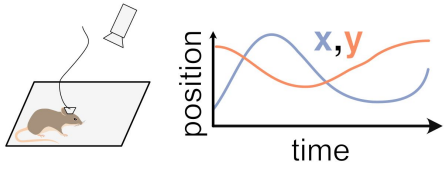
```
In [17]: np.concatenate((tsd1, tsd2))
Out[17]:
Time (s)
----------  ----------
0            0.910503
1           -0.0110368
2            0.496159
3           -0.956014
4           -0.592748
5           -0.711171
6           -0.370642
7            0.424684
8            0.718995
9            0.616419
10           0.159056
11           1.10691
12           0.856208
13           1.44663
14          -2.11429
15          -1.01082
16           0.563591
17           2.09225
18           0.484394
19           0.482061
dtype: float64, shape: (20,)
```

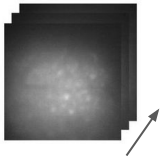
```
In [18]: np.concatenate((tsd2, tsd1))
-----------------------------------------
RuntimeError
Cell In[18], line 1
----> 1 np.concatenate((tsd2, tsd1))
```

```
RuntimeError: The order of the Tsd index sho
uld be strictly increasing and non overlappi
ng.
```

Time series without data : the timestamps object







FLATIRON INSTITUTE

Tsd: 1-dimension

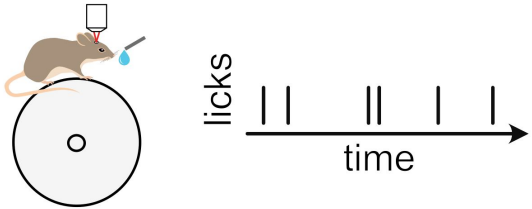
```
In [11]: tsd = nap.Tsd(t=t, d=d)
```

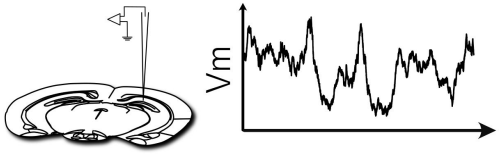
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])
```

TsdTensor: n-dimensions

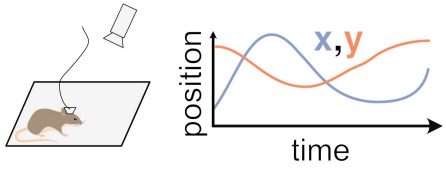
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```





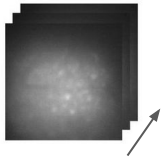
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



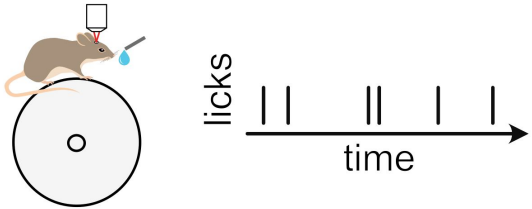
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])
```



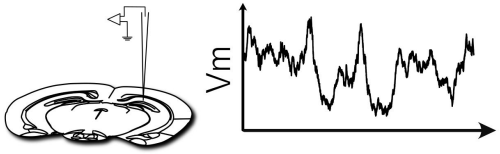
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



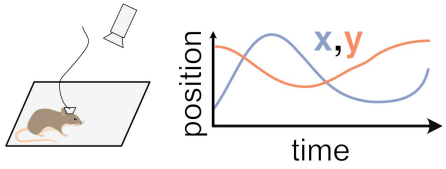
Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```



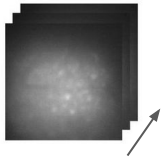
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



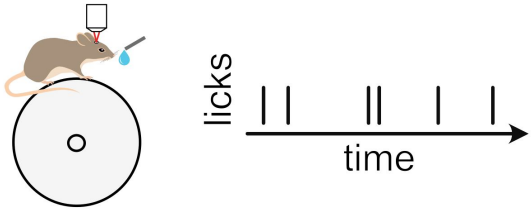
TsdFrame: 2-dimensions

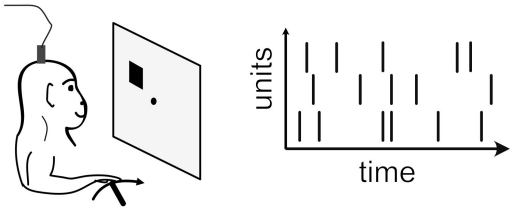
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                 columns = ['x', 'y'])
```



TsdTensor: n-dimensions

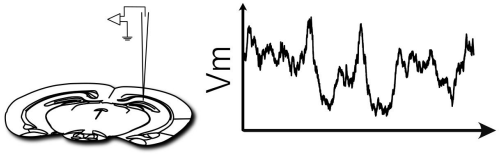
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```





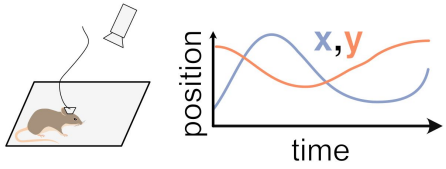
Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```



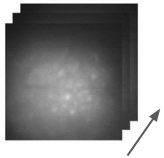
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



TsdFrame: 2-dimensions

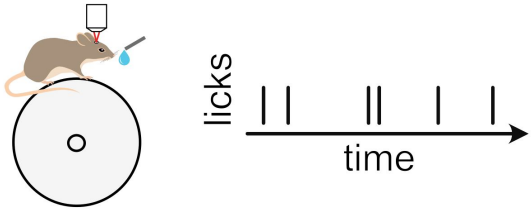
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])
```



TsdTensor: n-dimensions

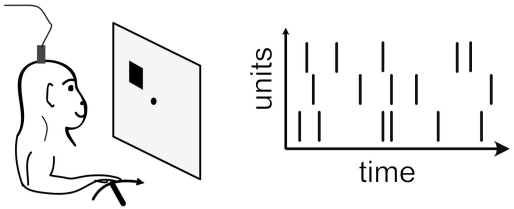
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```

FLATIRON INSTITUTE



Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```

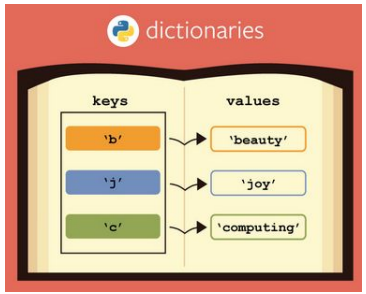


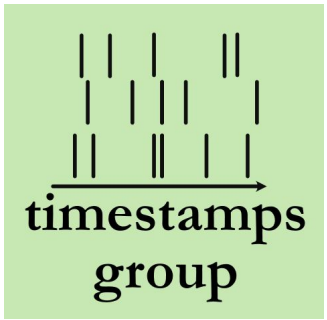
TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
Out[18]:
  Index    rate
-------  ------
      0   10.02
      1    5.01
      2    2.06
```

Collection of timestamps: TsGroup object

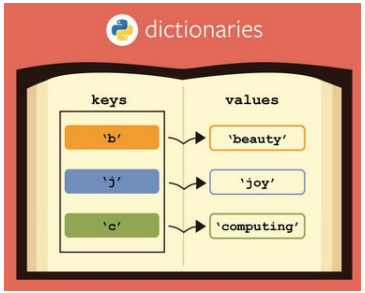
TsGroup manipulation

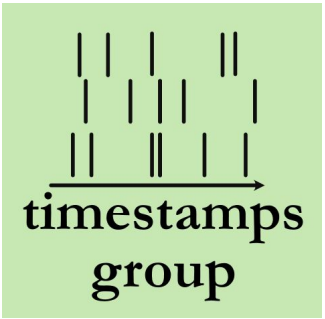




TsGroup manipulation

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    })
```

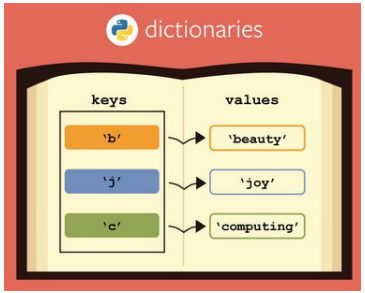


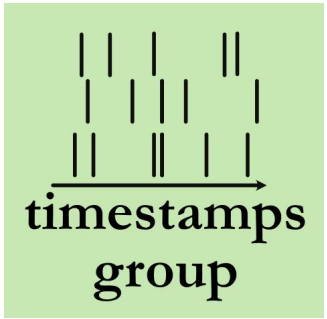


TsGroup manipulation

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    })
```

```
In [9]: ts_group
Out[9]:
  Index     rate
-------  -------
      0    1.001
      1   10.01
      2  100.1
```



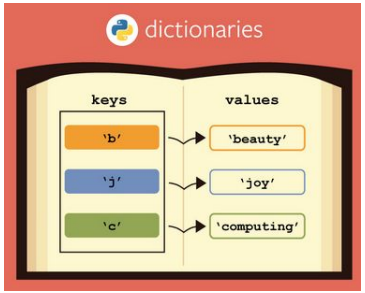


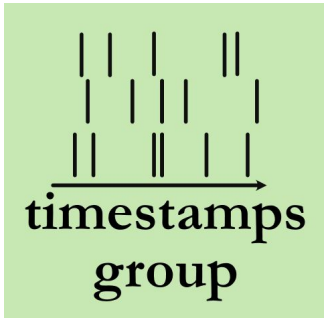
TsGroup manipulation

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    })
```

```
In [9]: ts_group
Out[9]:
  Index     rate
-------  -------
      0    1.001
      1   10.01
      2  100.1
```

```
In [10]: ts_group[[0, 2]]
Out[10]:
  Index     rate
-------  -------
      0    1.001
      2  100.1
```





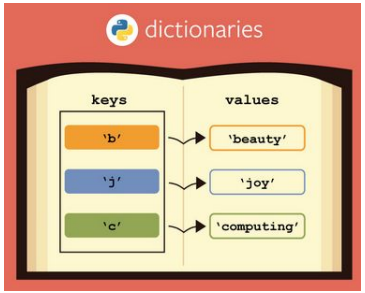
Operations :

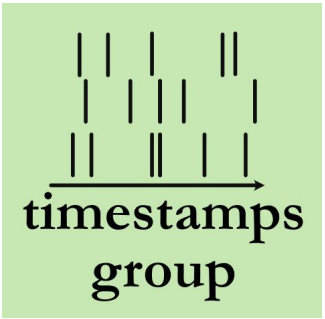
- restrict
- Binning
- …

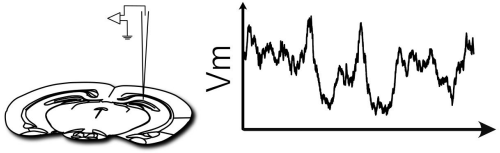
```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    })
```

```
In [9]: ts_group
Out[9]:
  Index      rate
-------  -------
      0     1.001
      1    10.01
      2   100.1
```

```
In [11]: ts_group.count(bin_size=2, time_units="s"
    ...: )
Out[11]:
Time (s)      0    1    2
----------  ---  ---  ---
1             2   20  200
3             2   20  200
5             2   20  200
7             2   20  200
9             2   20  200
dtype: float64, shape: (5, 3)
```

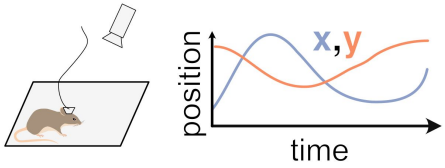






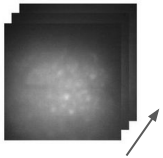
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



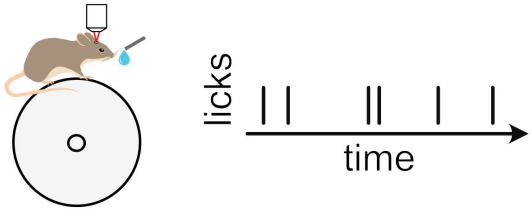
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])
```



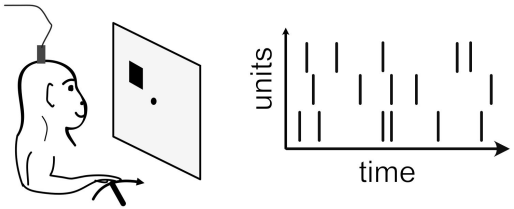
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



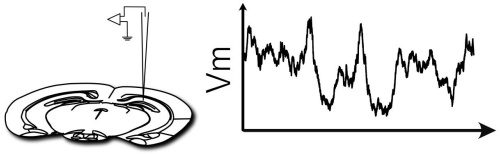
Ts: Timestamps

```
In [6]: nap.Ts(t)
```



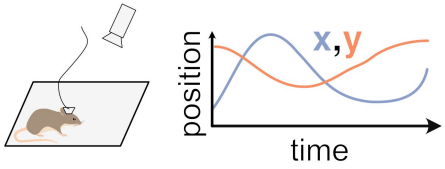
TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```



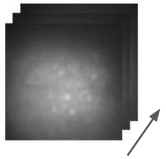
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



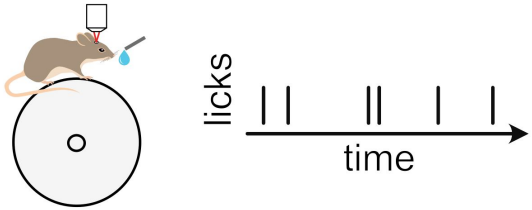
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                         columns = ['x', 'y'])
```



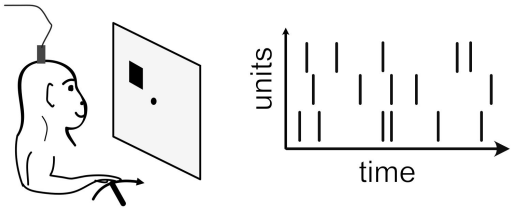
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



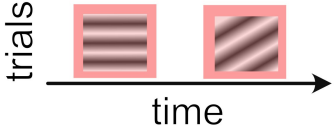
Ts: Timestamps

```
In [6]: nap.Ts(t)
```

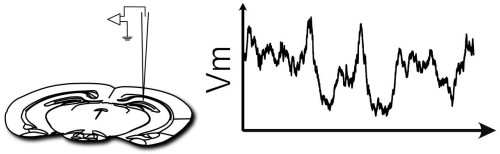


TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```

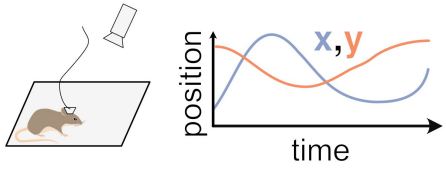


IntervalSet: set of epochs



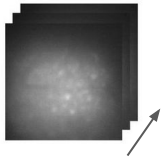
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



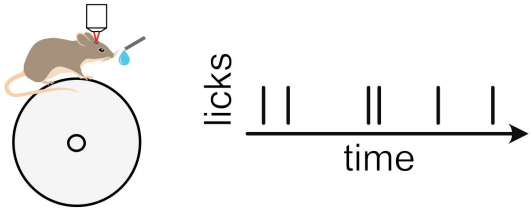
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                 columns = ['x', 'y'])
```



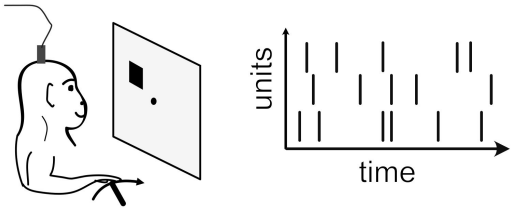
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



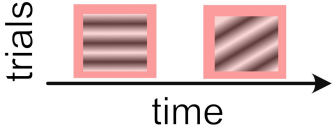
Ts: Timestamps

```
In [6]: nap.Ts(t)
```



TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```

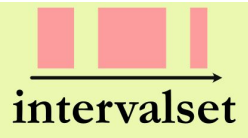


IntervalSet: set of epochs

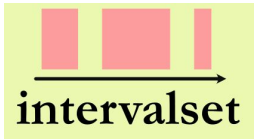
```
In [23]: nap.IntervalSet(
    ...:     start=start,
    ...:     end = end)
Out[23]:
   start   end
0    0.0   1.0
1    3.0   5.0
2    9.0  12.0
```

Intervals of time : the IntervalSet object

- Sleep/wake
- Stimulus on/off
- Lick start/end

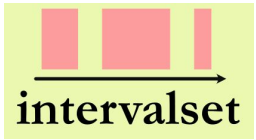


- Sleep/wake
- Stimulus on/off
- Lick start/end



	Start (second)	End (second)
Stim 0	0	1
Stim 1	3	5

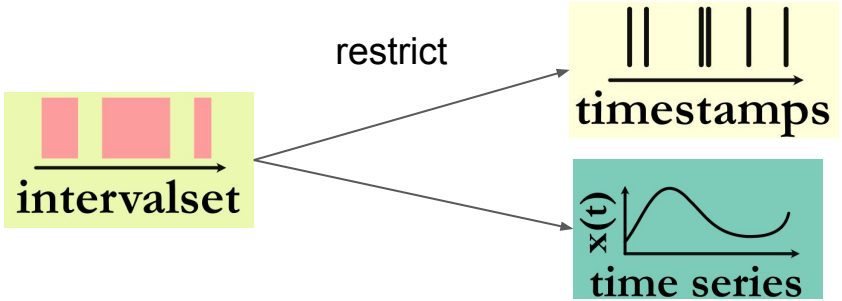
- Sleep/wake
- Stimulus on/off
- Lick start/end



	Start (second)	End (second)
Stim 0	0	1
Stim 1	3	5

```
In [26]: nap.IntervalSet(start=[0, 3], end=[1, 5])
Out[26]:
   start  end
0    0.0  1.0
1    3.0  5.0
```

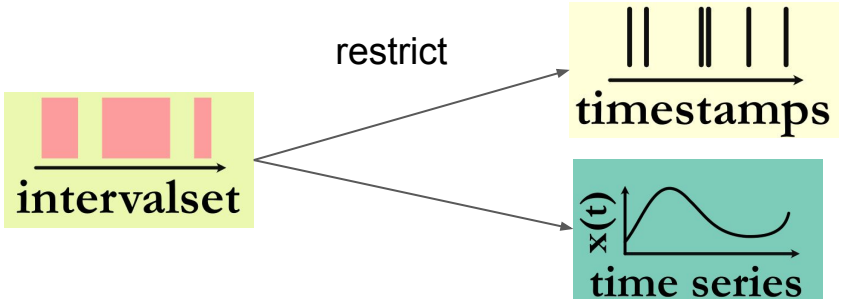
- Sleep/wake
- Stimulus on/off
- Lick start/end



	Start (second)	End (second)
Stim 0	0	1
Stim 1	3	5

```
In [26]: nap.IntervalSet(start=[0, 3], end=[1, 5])
Out[26]:
   start  end
0    0.0  1.0
1    3.0  5.0
```

- Sleep/wake
- Stimulus on/off
- Lick start/end

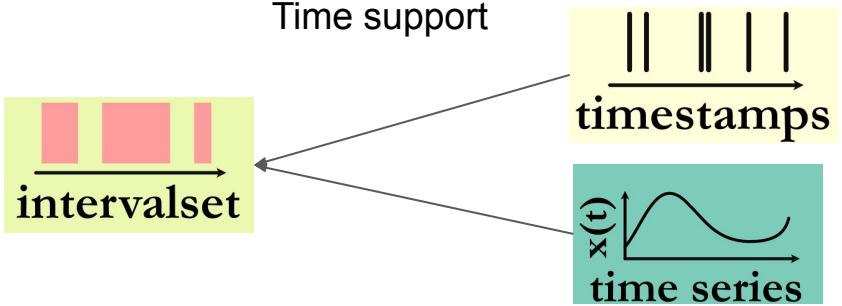


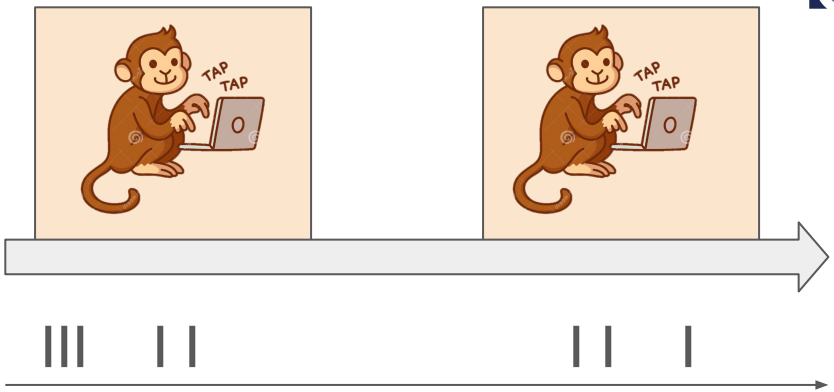

```
In [9]: ts
Out[9]:
Time (s)
0.36788271
0.664444232
1.188544252
1.482295474
5.058304181
5.119544635
5.412305372
7.032828073
7.85118454
8.561290173
shape: 10
```

```
In [6]: ep
Out[6]:
            start    end
        0       0      1
        1       3      5
shape: (2, 2), time unit: sec.
```

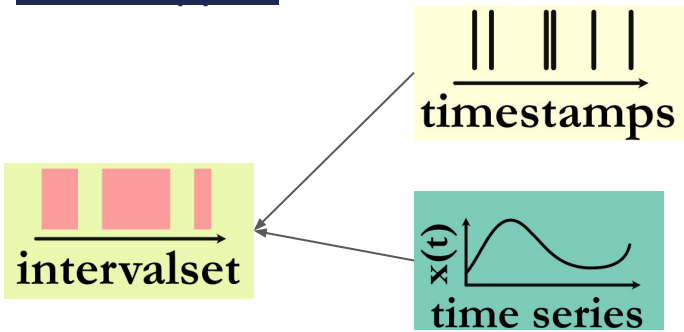
```
In [10]: ts.restrict(ep)
Out[10]:
Time (s)
0.36788271
0.664444232
shape: 2
```

Time support

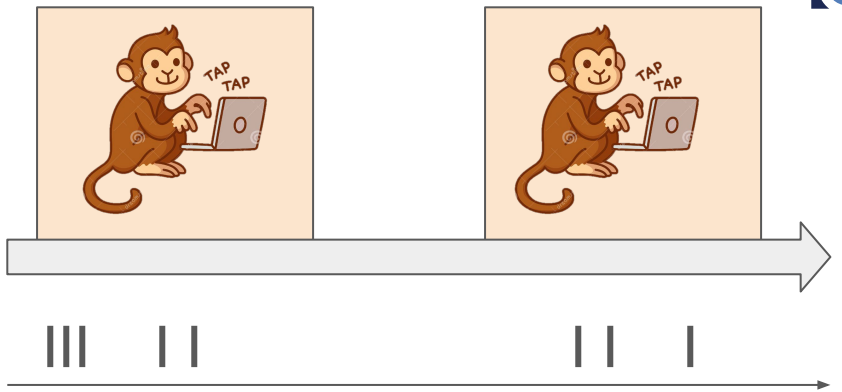




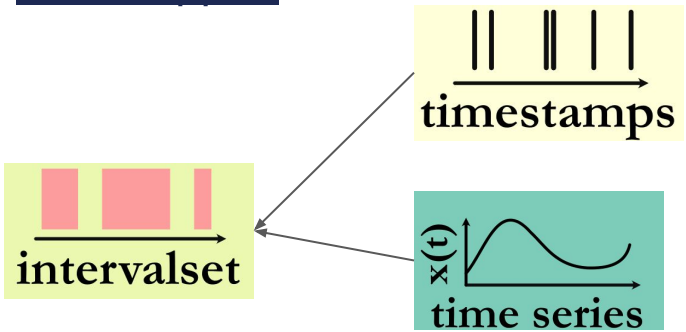
Time support



```
In [30]: tweeting_monkey
Out[30]:
Time (s)
0.126937916
0.320358519
0.707764437
1.942286662
3.163258247
...
96.895533151
97.002109352
98.01970871
98.842008394
99.609694697
shape: 100
```

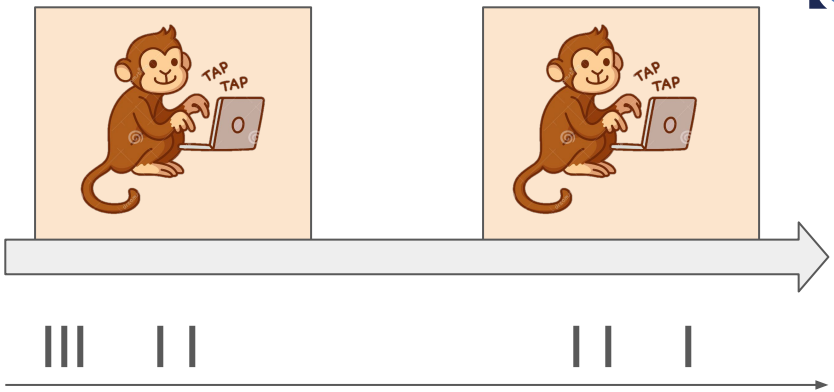


Time support

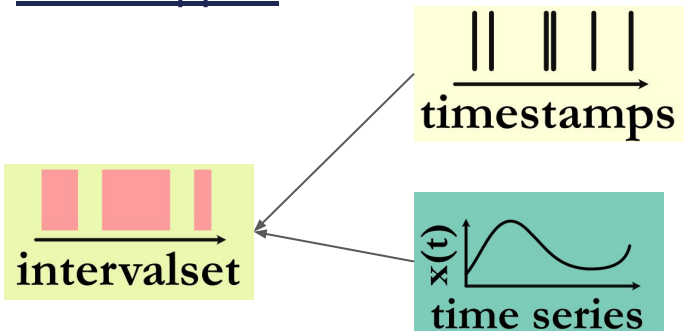


Tweet frequency?

```
In [30]: tweeting_monkey
Out[30]:
Time (s)
0.126937916
0.320358519
0.707764437
1.942286662
3.163258247
...
96.895533151
97.002109352
98.01970871
98.842008394
99.609694697
shape: 100
```

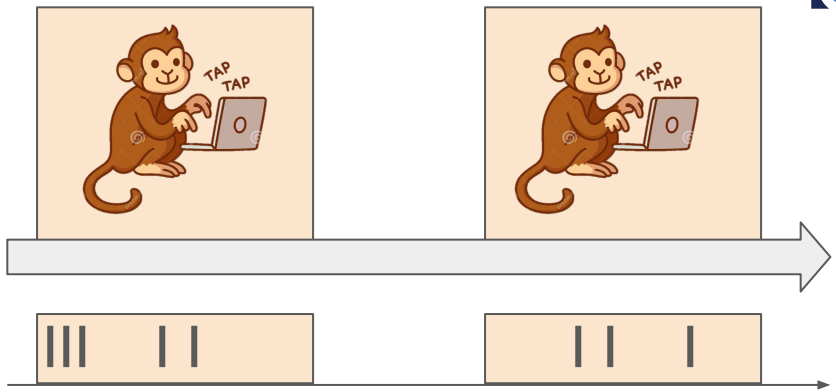


Time support

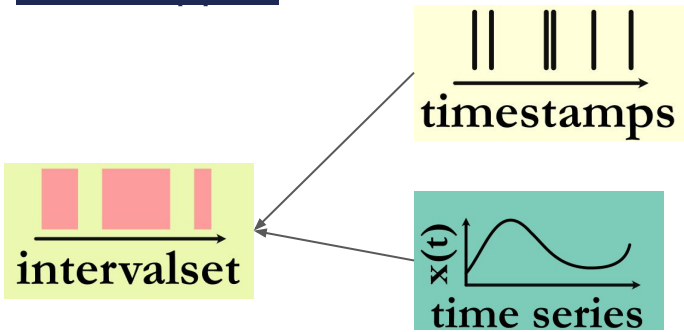


Tweet frequency?

```
In [30]: tweeting_monkey
Out[30]:
Time (s)
0.126937916
0.320358519
0.707764437
1.942286662
3.163258247
...
96.895533151
97.002109352
98.01970871
98.842008394
99.609694697
shape: 100
```

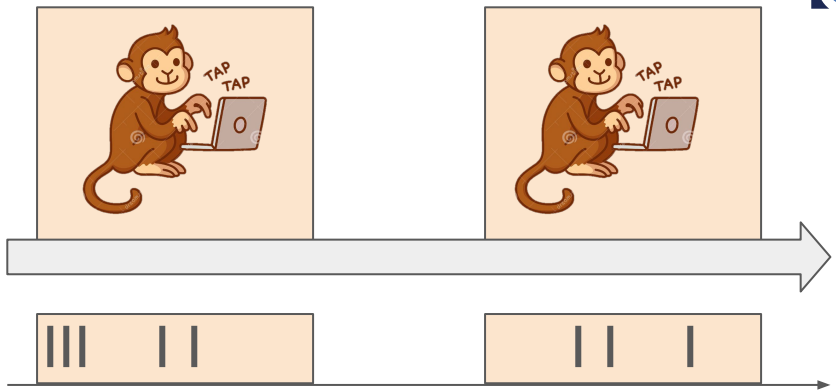


Time support



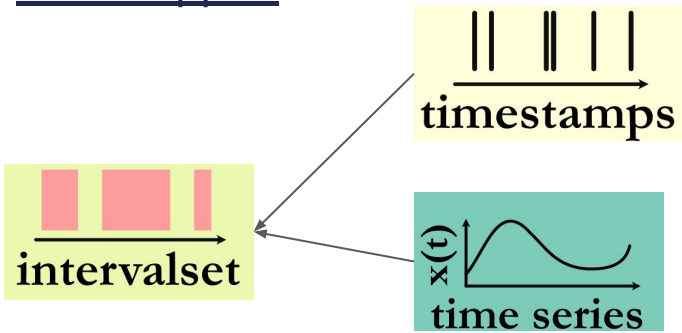
Tweet frequency?

```
In [30]: tweeting_monkey
Out[30]:
Time (s)
0.126937916
0.320358519
0.707764437
1.942286662
3.163258247
...
96.895533151
97.002109352
98.01970871
98.842008394
99.609694697
shape: 100
```



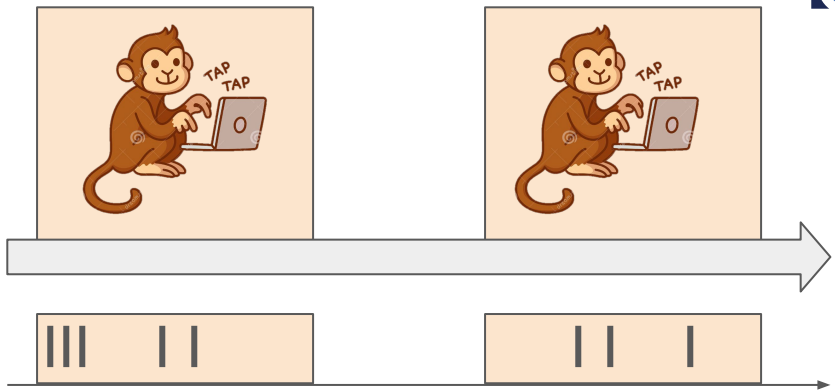

```
In [42]: tweeting_monkey.time_support
Out[42]:
   start    end
0    0.0   40.0
1   60.0  100.0
```

Time support



Tweet frequency?

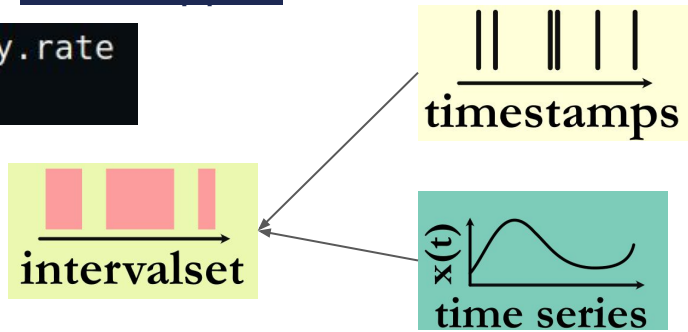
```
In [30]: tweeting_monkey
Out[30]:
Time (s)
0.126937916
0.320358519
0.707764437
1.942286662
3.163258247
...
96.895533151
97.002109352
98.01970871
98.842008394
99.609694697
shape: 100
```



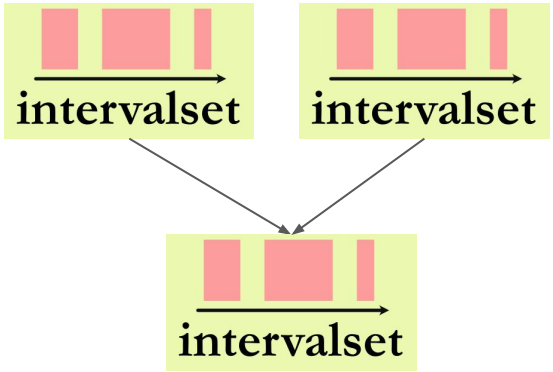

```
In [43]: tweeting_monkey.rate
Out[43]: 0.8125
```

```
In [42]: tweeting_monkey.time_support
Out[42]:
   start    end
0    0.0   40.0
1   60.0  100.0
```

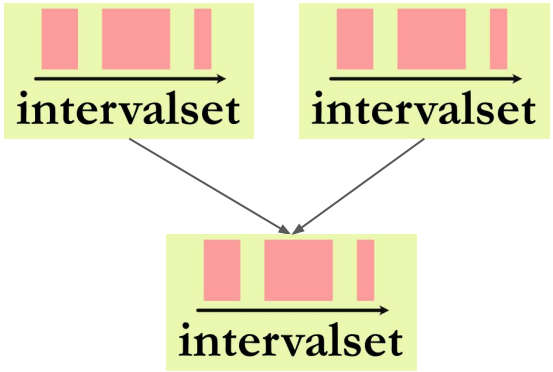
Time support



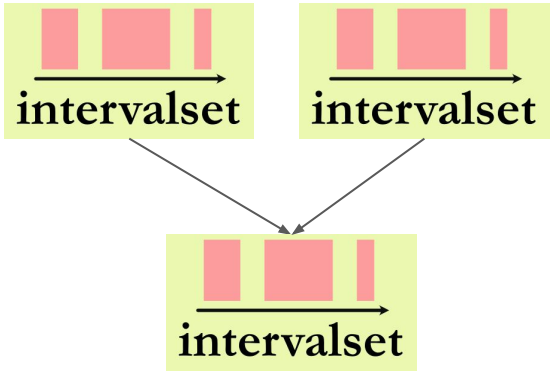
IntervalSet operations



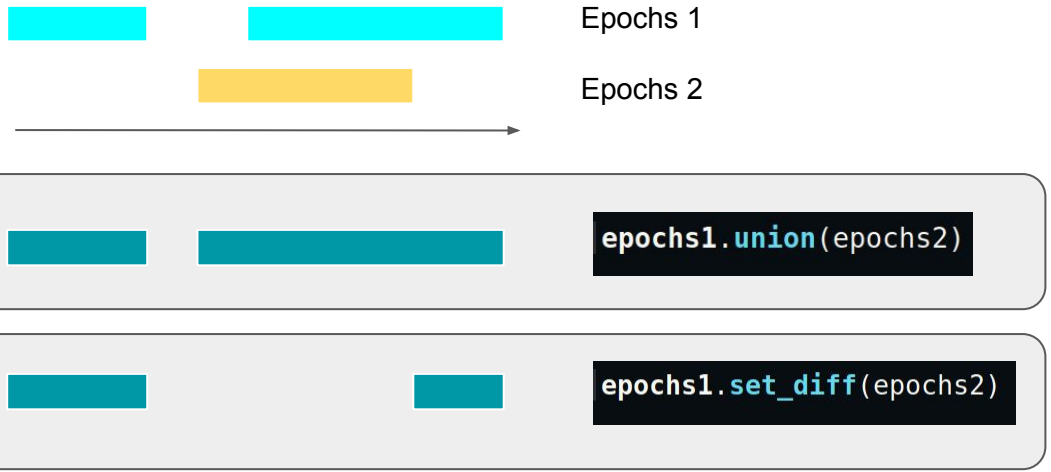
IntervalSet operations

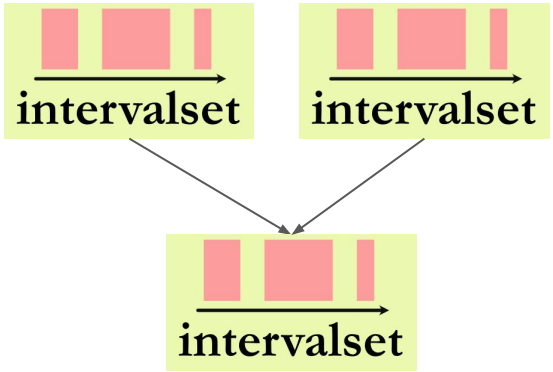


IntervalSet operations

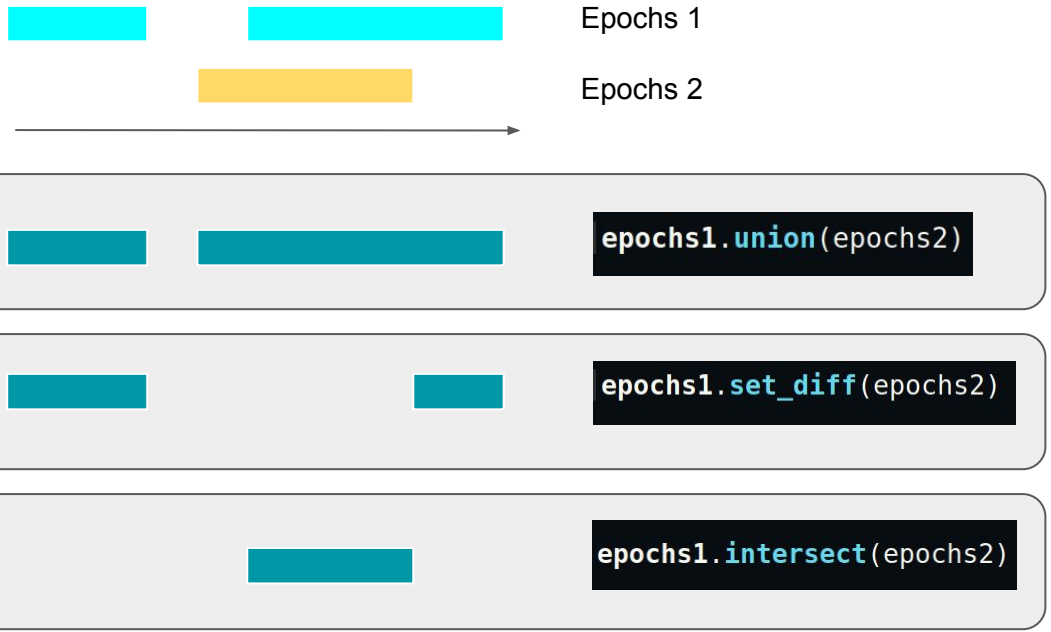


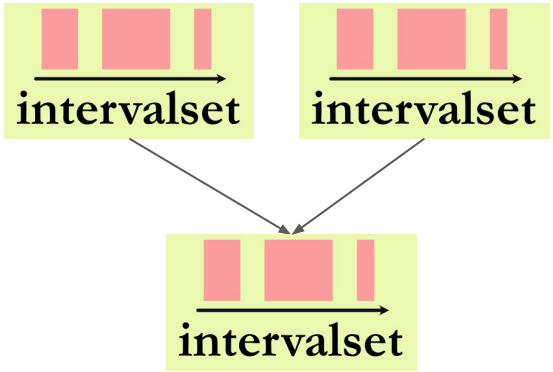
IntervalSet operations





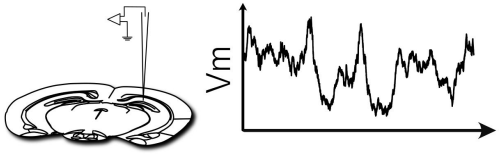
IntervalSet operations





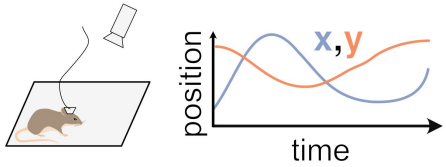
FLATIRON INSTITUTE

Summary : 6 objects to represent data



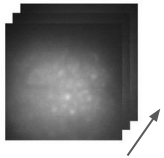
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



TsdFrame: 2-dimensions

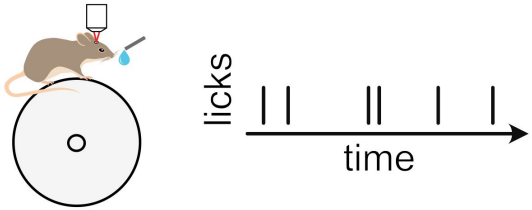
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:                 columns = ['x', 'y'])
```



TsdTensor: n-dimensions

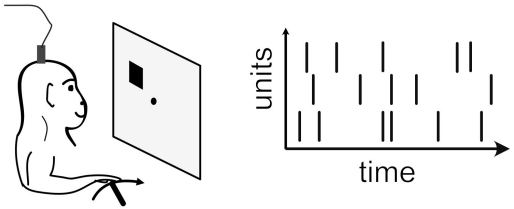
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```

FLATIRON INSTITUTE



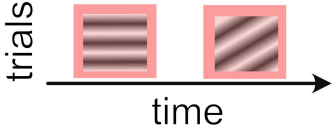
Ts: Timestamps

```
In [6]: nap.Ts(t)
```



TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```

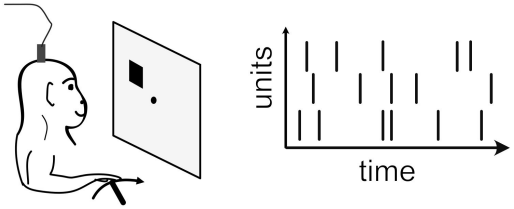


IntervalSet: set of epochs

```
In [23]: nap.IntervalSet(
    ...:             start=start,
    ...:             end = end)
```

FLATIRON INSTITUTE

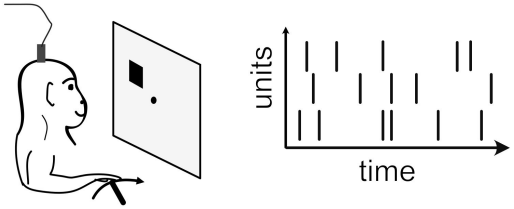
Metadata : extra informations



TsGroup: group of timestamps

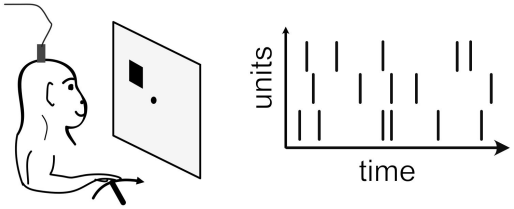
```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    })
```

```
In [9]: ts_group
Out[9]:
  Index      rate
-------  -------
      0     1.001
      1    10.01
      2   100.1
```



TsGroup: group of timestamps

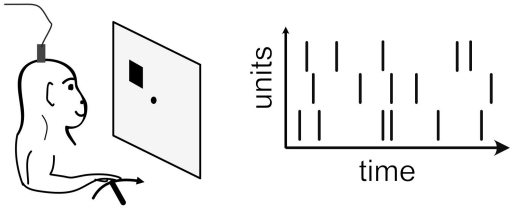
```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    },
    metadata={
        "location":["thalamus", "ca1", "cerebellum"]
        }
    )
```



TsGroup: group of timestamps

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    },
    metadata={
        "location":["thalamus", "ca1", "cerebellum"]
        }
    )
```

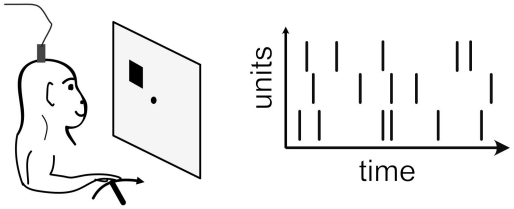
```
In [17]: ts_group
Out[17]:
  Index     rate  location
-------  -------  ----------
      0    1.001  thalamus
      1   10.01   ca1
      2  100.1    cerebellum
```



TsGroup: group of timestamps

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    },
    metadata={
        "location":["thalamus", "ca1", "cerebellum"],
        "direction":[0.1, 0.3, 0.2425]
        }
    )
```

```
In [35]: ts_group
Out[35]:
  Index     rate  location      direction
-------  -------  ----------  -----------
      0    1.001  thalamus          0.1
      1   10.01   ca1               0.3
      2  100.1    cerebellum        0.2425
```

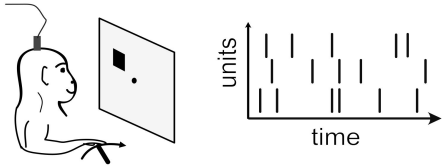


TsGroup: group of timestamps

```
ts_group = nap.TsGroup(
    data = {
        0: neuron_thalamus,
        1: neuron_ca1,
        2: neuron_cerebellum
    },
    metadata={
        "location":["thalamus", "ca1", "cerebellum"],
        "direction":[0.1, 0.3, 0.2425]
        }
    )
```

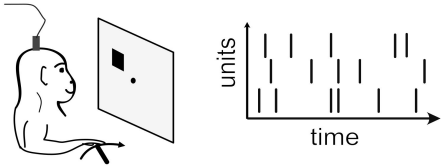
```
ts_group['alpha'] = np.random.randn(3)
ts_group.alpha = np.random.randn(3)
ts_group.set_info(alpha=np.random.randn(3))
```

```
In [22]: ts_group
Out[22]:
  Index      rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



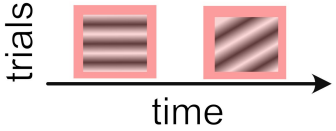
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

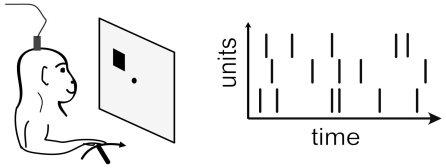


TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

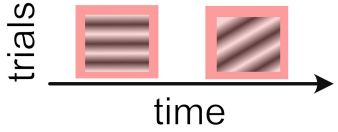


IntervalSet: set of epochs



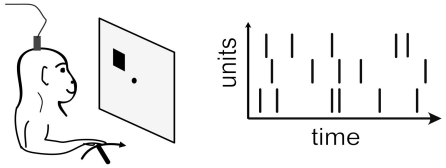
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



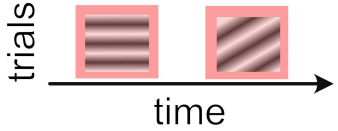
IntervalSet: set of epochs

```
iset = nap.IntervalSet(
    start = [0, 12, 26],
    end = [3, 19, 1890],
    metadata = {
        "trial_type":["left", "right", "left"]
        }
    )
```



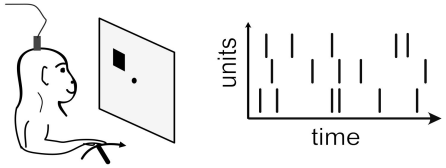
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



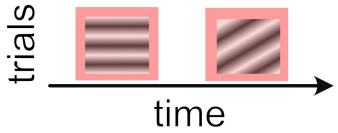
IntervalSet: set of epochs

```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```



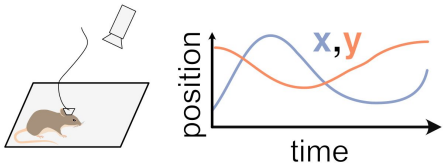
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

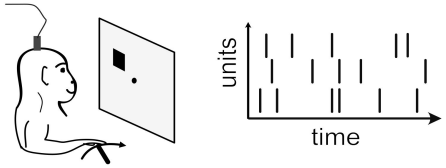


IntervalSet: set of epochs

```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```

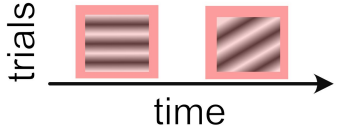


TsdFrame: 2-dimensions



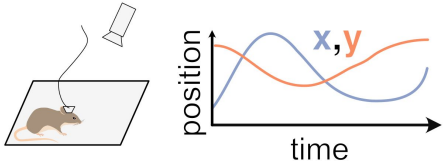
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



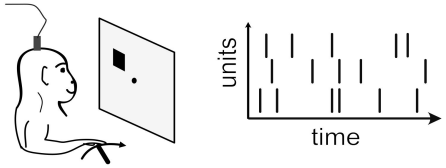
IntervalSet: set of epochs

```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```



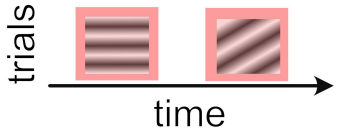
TsdFrame: 2-dimensions

```
tsdframe = nap.TsdFrame(
    t=np.arange(3),
    d=np.random.randn(3,2),
    columns=['x','y'],
    metadata = {
        "colors":["blue", "orange"]
        }
    )
```



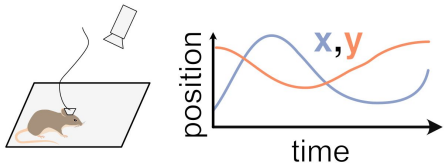
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



IntervalSet: set of epochs

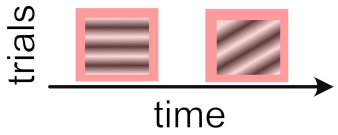
```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```



TsdFrame: 2-dimensions

```
In [36]: tsdframe
Out[36]:
Time (s)    x         y
----------  --------  --------
0.0         1.1911    1.08601
1.0         -2.53084  -0.39575
2.0         0.29567   0.00021
Metadata
--------    --------  --------
colors      blue      orange

dtype: float64, shape: (3, 2)
```

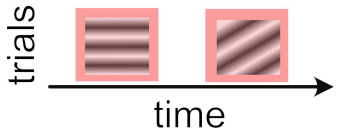
Accessing metadata



IntervalSet: set of epochs

```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```

Accessing metadata



IntervalSet: set of epochs

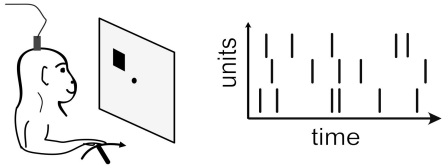
```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```

```
In [38]: iset.trial_type
Out[38]:
0     left
1    right
2     left
Name: trial_type, dtype: object
```

```
In [39]: iset['trial_type']
Out[39]:
0     left
1    right
2     left
Name: trial_type, dtype: object
```

```
In [42]: iset.metadata
Out[42]:
  trial_type
0       left
1      right
2       left
```

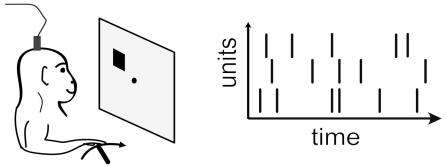
Slicing using metadata



TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

Slicing using metadata

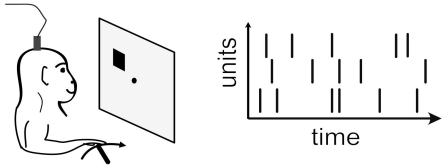


TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

```
In [46]: ts_group[ts_group.location == "thalamus"]
Out[46]:
  Index    rate  location      direction    alpha
-------  ------  ----------  -----------  -------
      0   1.001  thalamus            0.1  0.30967
```

Slicing using metadata

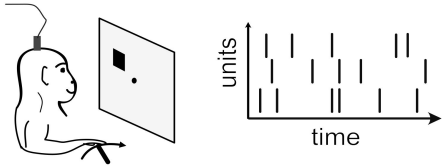


TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index     rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```

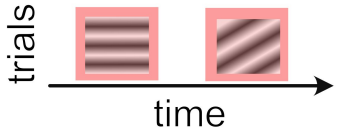
```
In [46]: ts_group[ts_group.location == "thalamus"]
Out[46]:
  Index    rate  location      direction    alpha
-------  ------  ----------  -----------  -------
      0   1.001  thalamus            0.1  0.30967
```

```
In [51]: ts_group[(ts_group.rate>5.0) & (ts_group.direction == 0.3)]
Out[51]:
  Index    rate  location      direction      alpha
-------  ------  ----------  -----------  ---------
      1   10.01  ca1                 0.3  -0.922495
```



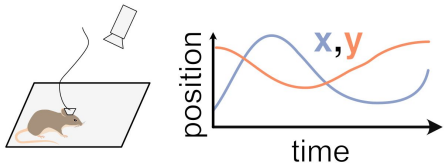
TsGroup: group of timestamps

```
In [22]: ts_group
Out[22]:
  Index      rate  location      direction      alpha
-------  -------  ----------  -----------  ---------
      0    1.001  thalamus          0.1      0.30967
      1   10.01   ca1               0.3     -0.922495
      2  100.1    cerebellum        0.2425  -0.482796
```



IntervalSet: set of epochs

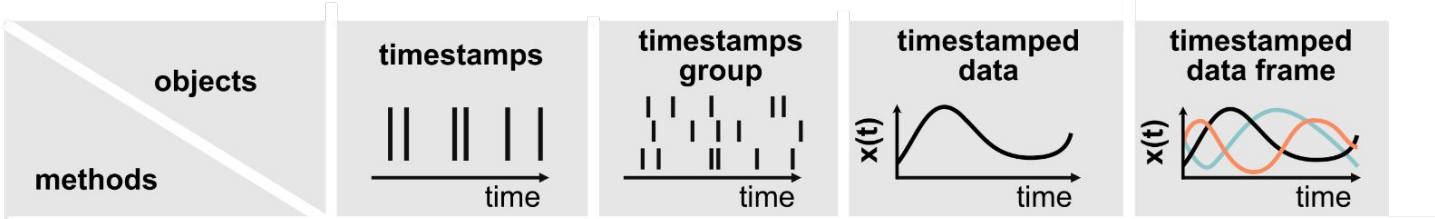
```
In [30]: iset
Out[30]:
  index    start    end  trial_type
      0        0      3  left
      1       12     19  right
      2       26   1890  left
shape: (3, 2), time unit: sec.
```



TsdFrame: 2-dimensions

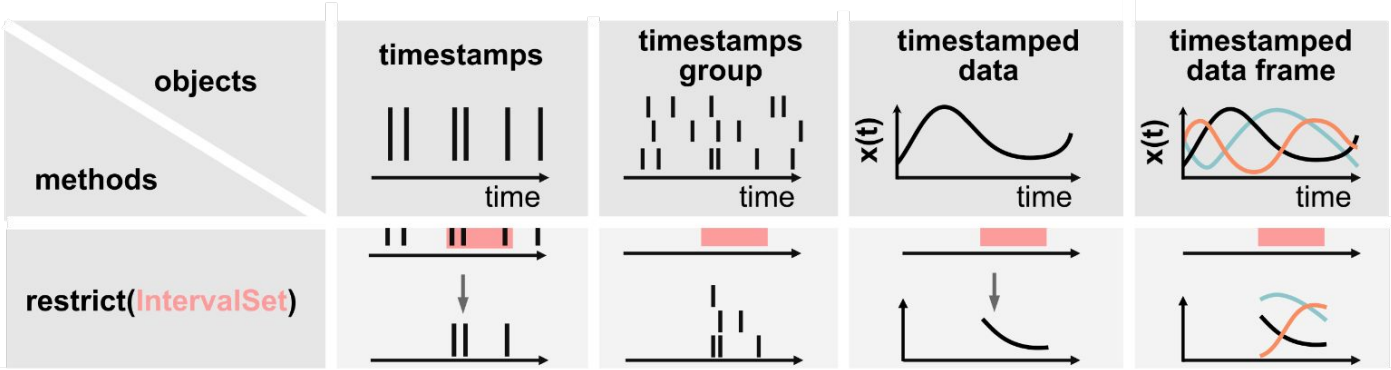
```
In [36]: tsdframe
Out[36]:
Time (s)    x         y
----------  --------  --------
0.0         1.1911    1.08601
1.0         -2.53084  -0.39575
2.0         0.29567   0.00021
Metadata
--------    --------  --------
colors      blue      orange

dtype: float64, shape: (3, 2)
```

Core functions of pynapple



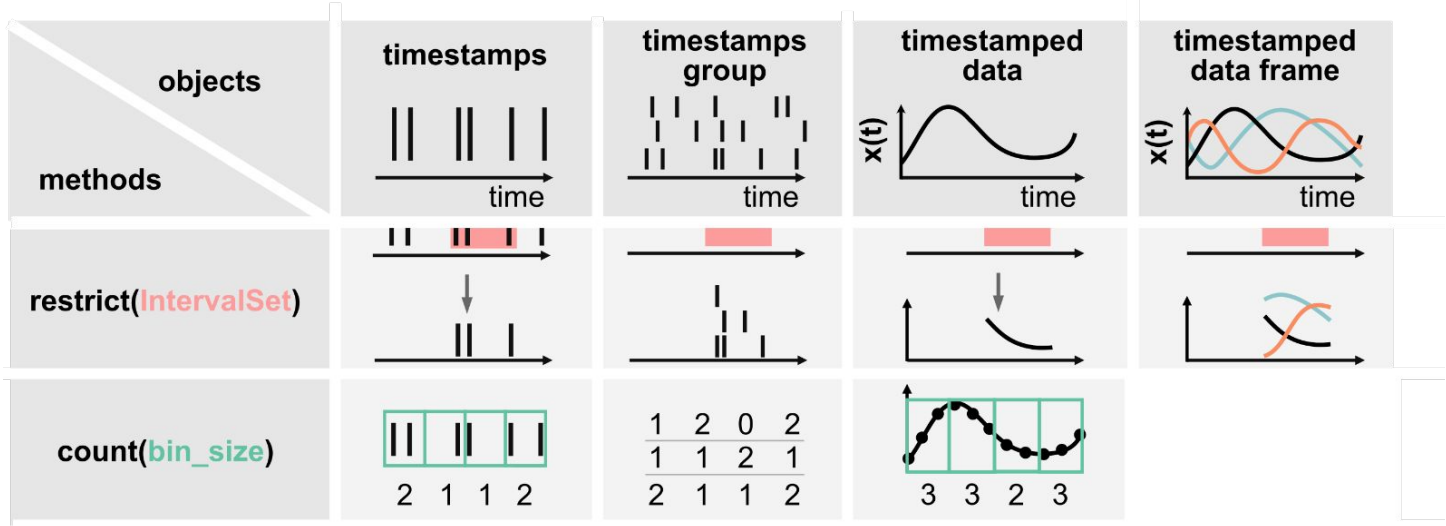
FLATIRON INSTITUTE

Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.

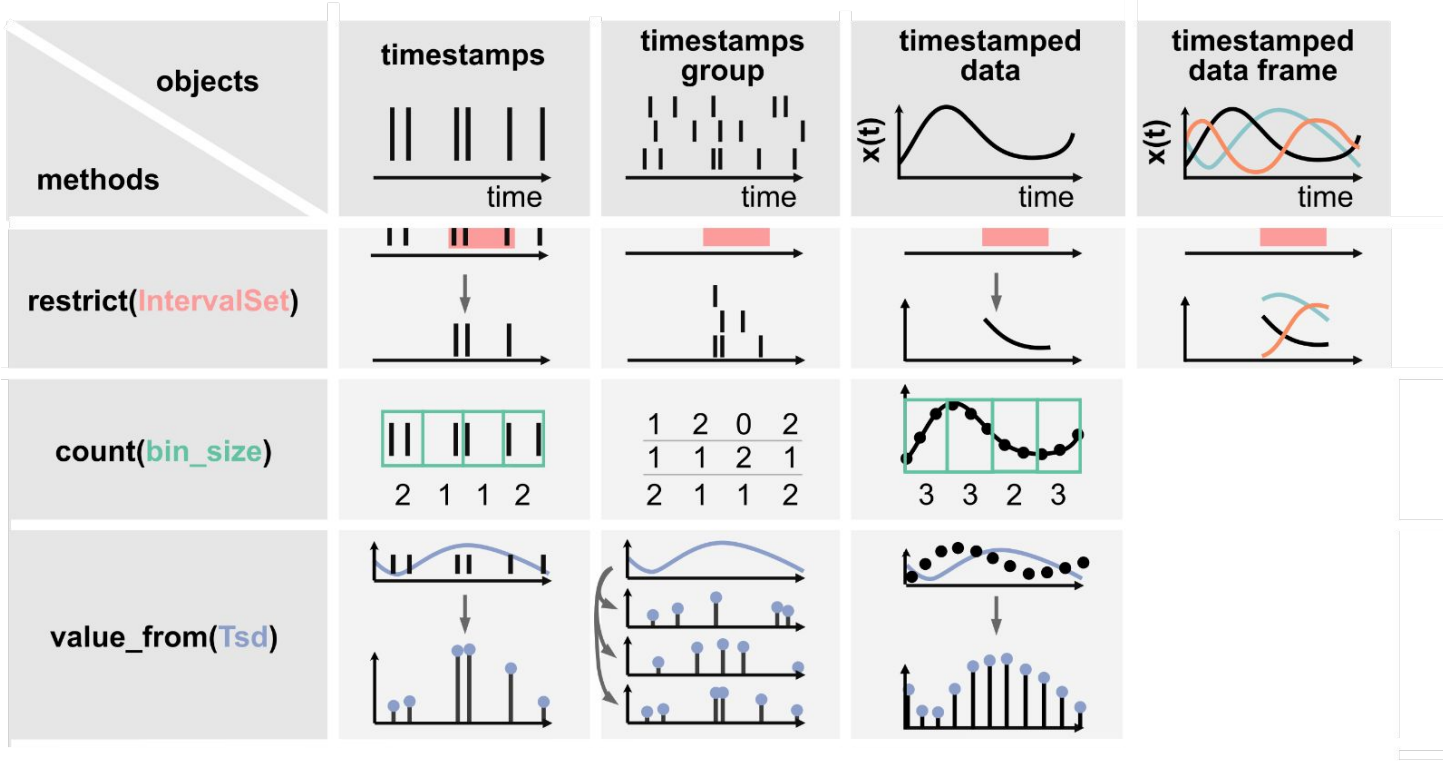


FLATIRON INSTITUTE

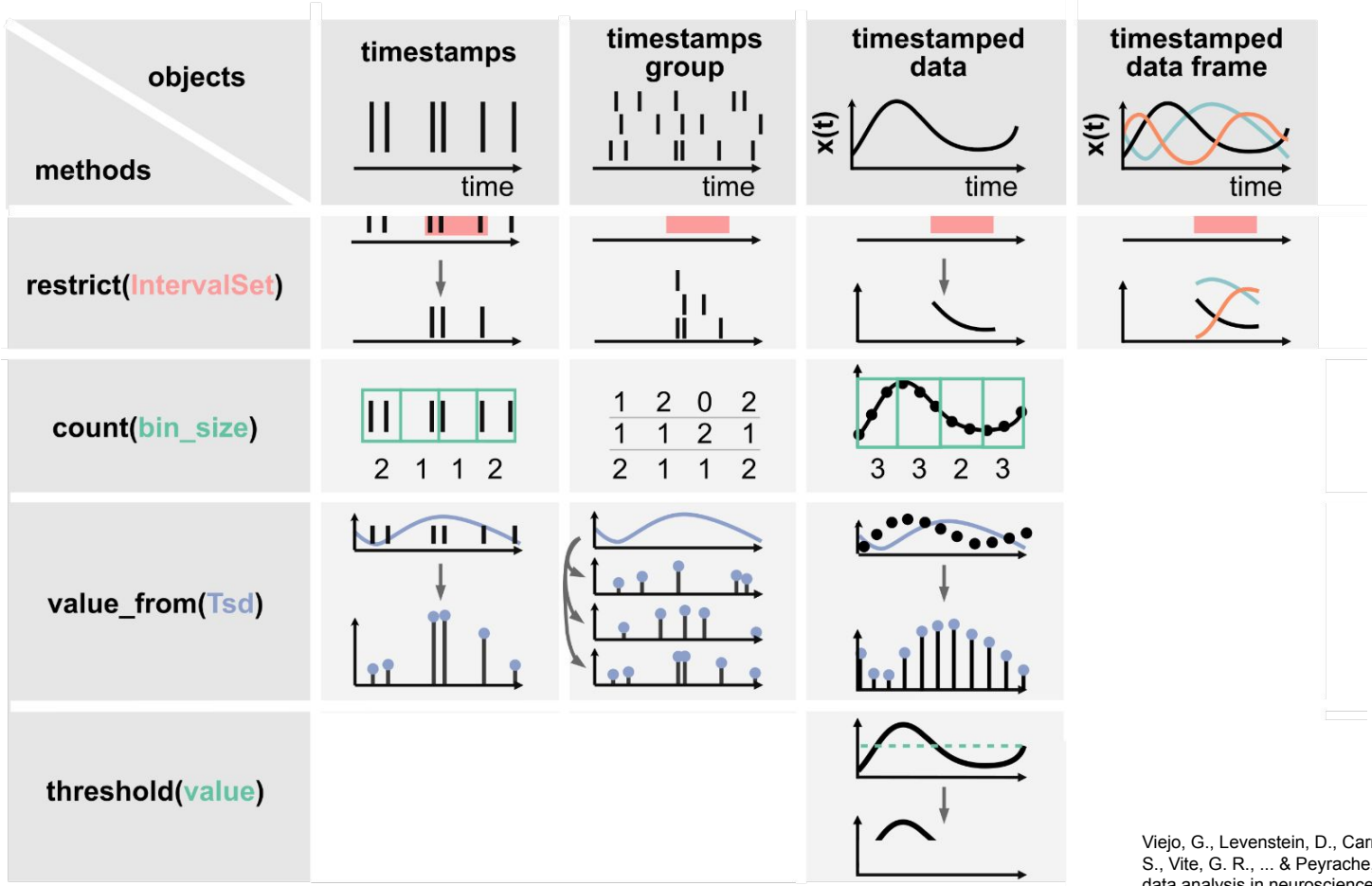
Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.



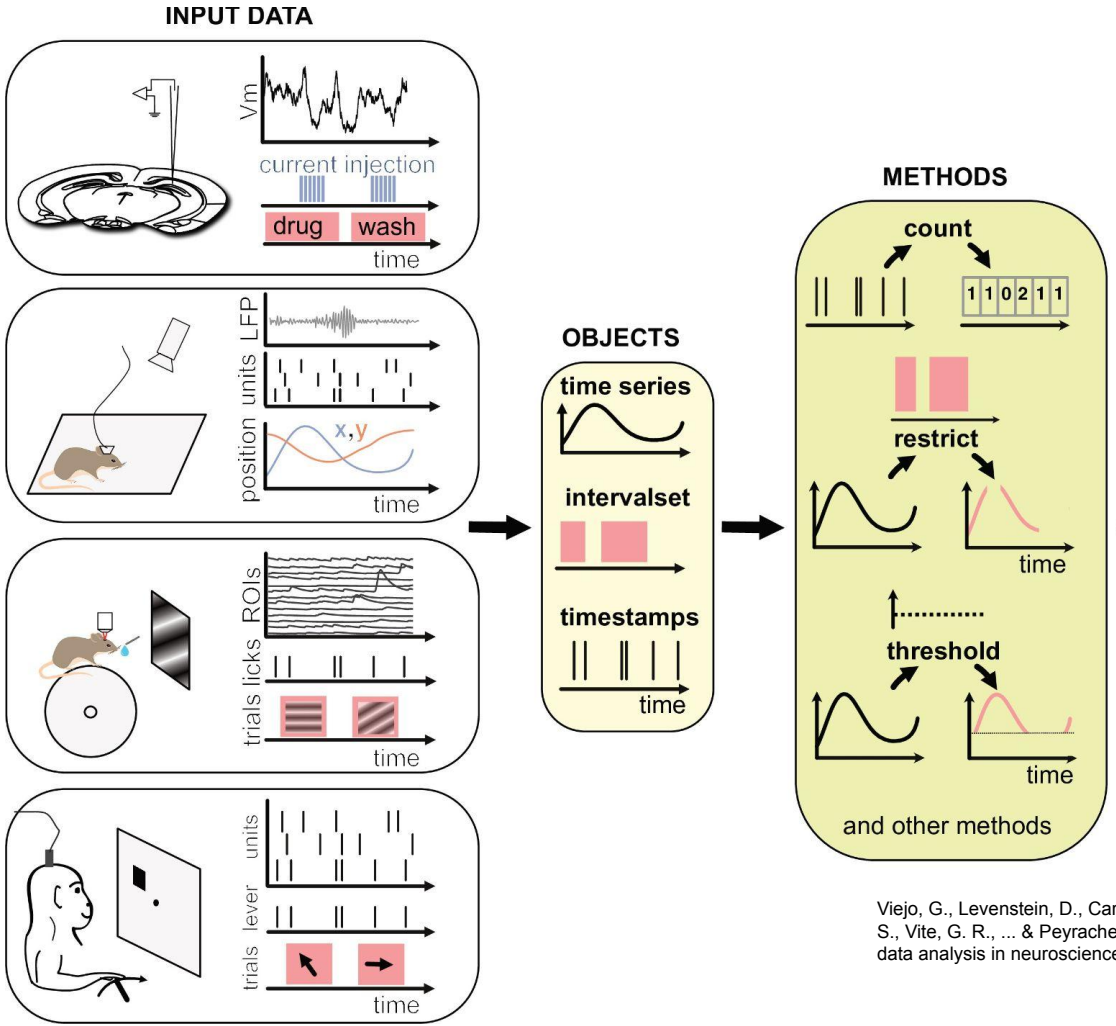
Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.

Pynapple IO

From **spikeinterface** : a unified framework for spike sorting

Raw Data Formats

For raw recording formats, we currently support:

- AlphaOmega `read_alphaomega()`
- Axona `read_axona()`
- BlackRock `read_blackrock()`
- Binary `read_binary()`
- Biocam HDF5 `read_biocam()`
- CED `read_ced()`
- EDF `read_edf()`
- IBL streaming `read_ibl_streaming_recording()`
- Intan `read_intan()`
- MaxWell `read_maxwell()`
- MCS H5 `read_mcsh5()`
- MCS RAW `read_mcsraw()`
- MEArec `read_mearec()`
- Mountainsort MDA `read_mda_recording()`
- Neuralynx `read_neuralynx()`
- Neurodata Without Borders `read_nwb_recording()`
- Neuroscope `read_neuroscope_recording()`
- Neuroexplorer `read_neuroexplorer()`
- NIX `read_nix()`
- Open Ephys Legacy `read_openephys()`
- Open Ephys Binary `read_openephys()`
- Plexon `read_plexon()`
- Plexon 2 `read_plexon2()`
- Shybrid `read_shybrid_recording()`
- SpikeGLX `read_spikeglx()`
- SpikeGLX IBL compressed `read_cbin_ibl()`
- SpikeGLX IBL stream `read_streaming_ibl()`
- Spike 2 `read_spike2()`
- TDT `read_tdt()`
- Zarr `read_zarr()`

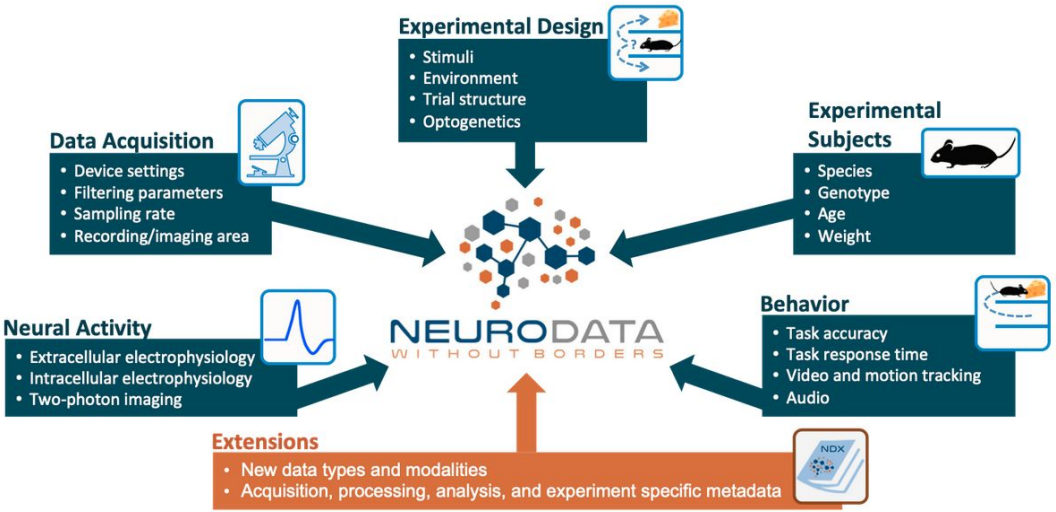
Sorted Data Formats

For sorted data formats, we currently support:

- BlackRock `read_blackrock_sorting()`
- Combinato `read_combinato()`
- Cell explorer `read_cellexplorer()`
- HerdingSpikes2 `read_herdingspikes()`
- HDsort `read_hdsort()`
- Kilosort1/2/2.5/3 `read_kilosort()`
- Klusta `read_klusta()`
- MClust `read_mclust()`
- MEArec `read_mearec()`
- Mountainsort MDA `read_mda_sorting()`
- Neurodata Without Borders `read_nwb_sorting()`
- Neuroscope `read_neuroscope_sorting()`
- Neuralynx spikes `read_neuralynx_sorting()`
- NPZ (created by SpikeInterface) `read_npz_sorting()`
- Plexon spikes `read_plexon_sorting()`
- Plexon 2 spikes `read_plexon2_sorting()`
- Shybrid `read_shybrid_sorting()`
- Spyking Circus `read_spykingcircus()`
- Trideclous `read_tridesclous()`
- Wave Clus `read_waveclus()`
- YASS `read_yass()`

On universal standard : the neurodata without borders format (NWB)

Neurodata Without Borders (NWB) is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build common analysis tools for neurophysiology data.



Teeters, J. L., Godfrey,K., Young, R., Dang, C., Friedsam, C., Wark, B., ... & Sommer, F. T. (2015). Neurodata without borders: creating a common data format for neurophysiology. *Neuron*, *88*(4), 629-634.

Rübel, O., Tritt, A., Ly, R., Dichter, B. K., Ghosh, S., Niu, L., ... & Bouchard, K. E. (2022). The neurodata without borders ecosystem for neurophysiological data science. *Elife*, *11*, e78362.

Loading NWB with pynapple

NWB interface

In [1]:

```
1 import pynapple as nap
2
3 nwb = nap.load_file("A2929-200711.nwb")
```

FLATIRON INSTITUTE

NWB interface

In [1]:
```
1 import pynapple as nap
2
3 nwb = nap.load_file("A2929-200711.nwb")
```

In [2]:
```
1 nwb
```

A2929-200711.nwb

Keys	Type
position_time_support	IntervalSet
epochs	IntervalSet
z	Tsd
y	Tsd
x	Tsd
rz	Tsd
ry	Tsd
rx	Tsd

NWB interface

In [1]:
```
import pynapple as nap

nwb = nap.load_file("A2929-200711.nwb")
```

In [2]:
```
nwb
```

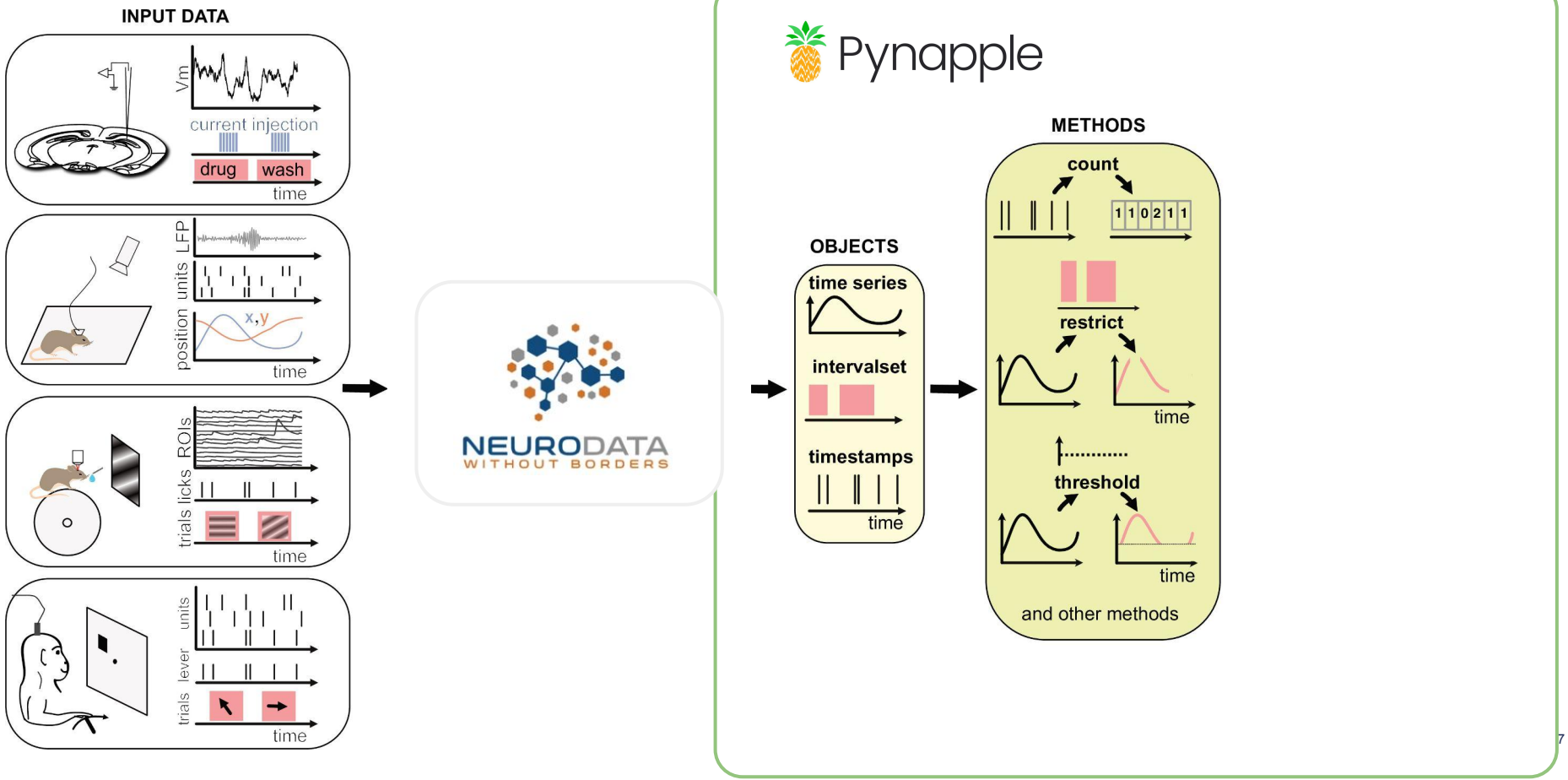
A2929-200711.nwb

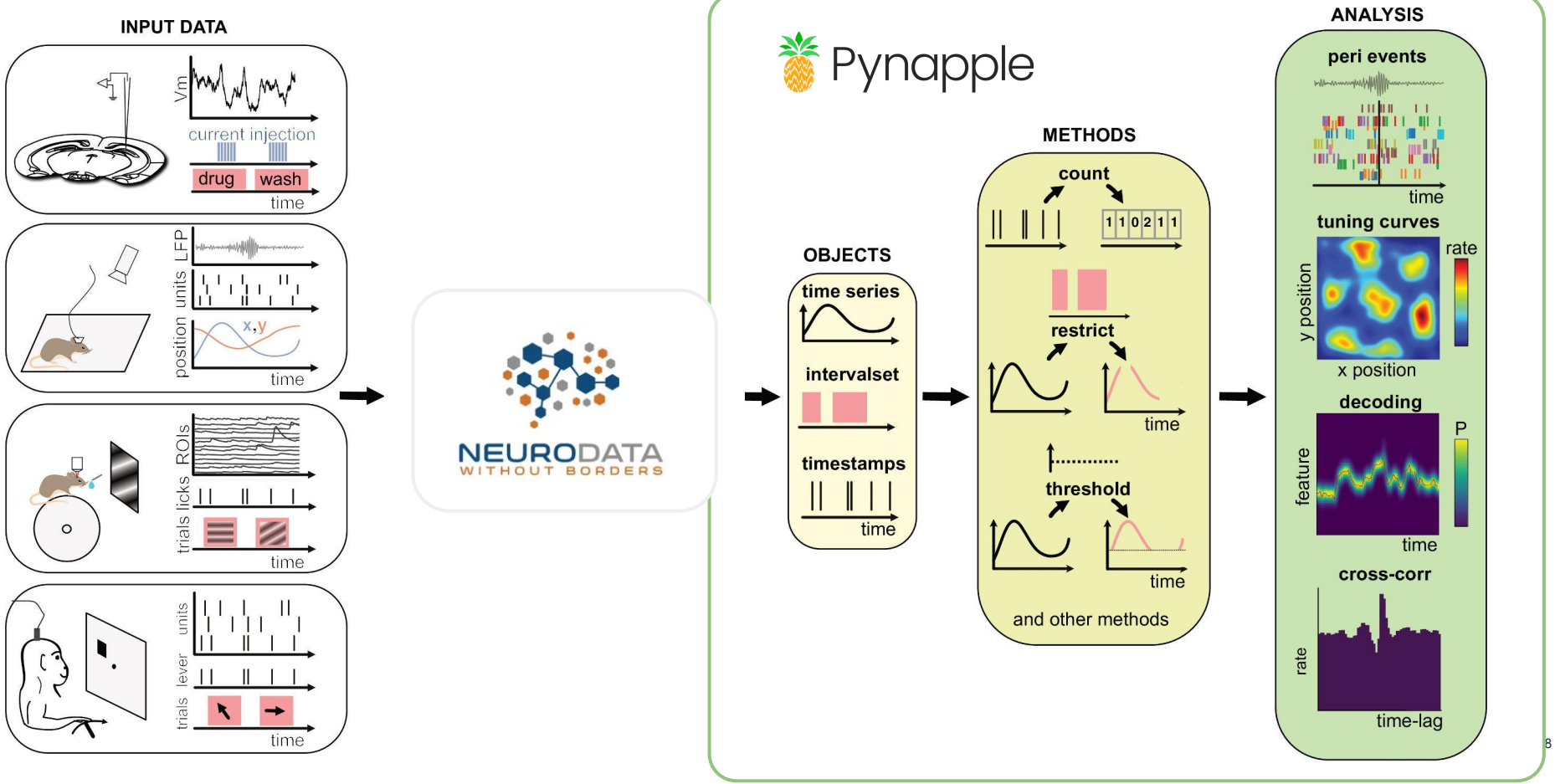
Keys	Type
position_time_support	IntervalSet
epochs	IntervalSet
z	Tsd
y	Tsd
x	Tsd
rz	Tsd
ry	Tsd
rx	Tsd

In [3]:
```
nwb["position_time_support"]
```

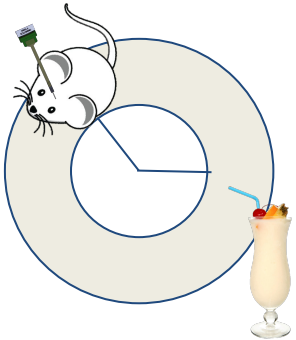
Out[3]:

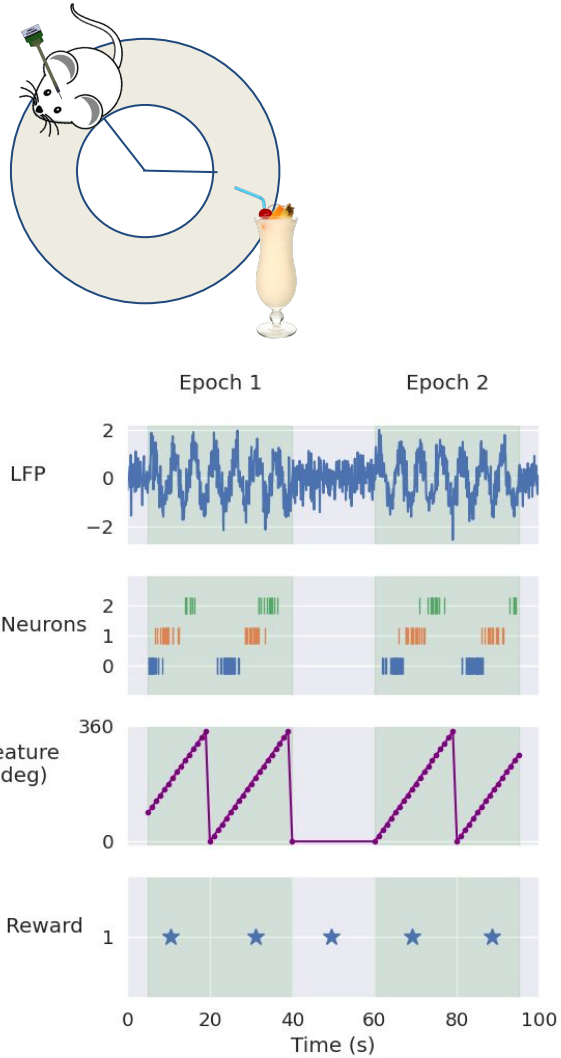
	start	end
0	670.6407	1199.99495

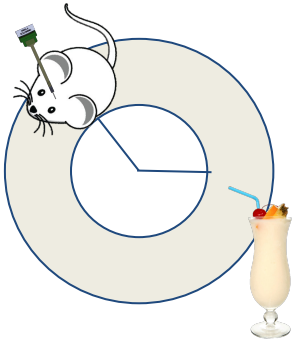

INPUT DATA
Vm
current injection
drug
wash
time
LFP
units
position
x,y
time
ROIs
licks
trials
time
units
lever
trials
time
NEURODATA
WITHOUT BORDERS
Pynapple
OBJECTS
time series
intervalset
timestamps
time
METHODS
count
1 1 0 2 1 1
restrict
time
threshold
time
and other methods


INPUT DATA
Vm
current injection
drug
wash
time
LFP
units
position
x,y
time
ROIs
licks
trials
time
units
lever
trials
time
NEURODATA
WITHOUT BORDERS
Pynapple
OBJECTS
time series
intervalset
timestamps
time
METHODS
count
1 1 0 2 1 1
restrict
time
threshold
time
and other methods
ANALYSIS
peri events
time
tuning curves
rate
y position
x position
decoding
P
feature
time
cross-corr
rate
time-lag

Standard analysis in systems neuroscience (Tomorrow)

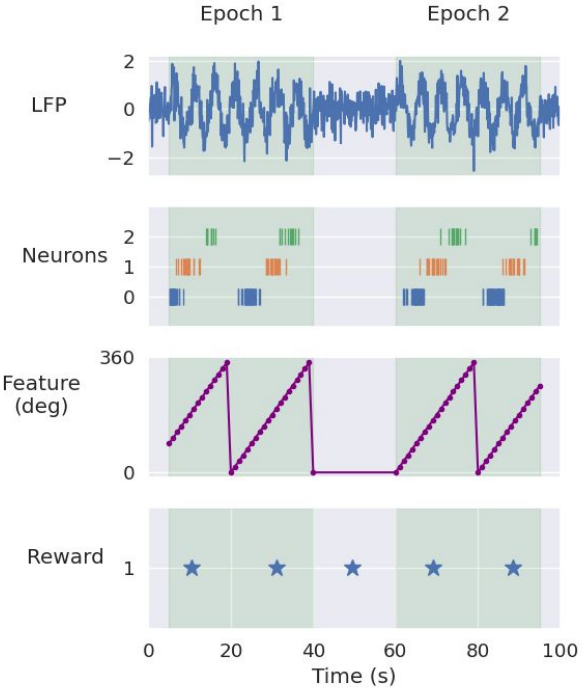



Epoch 1
Epoch 2
LFP
2
0
−2
Neurons
2
1
0
Feature
(deg)
360
0
Reward
1
0
20
40
60
80
100
Time (s)



```
In [48]: ep
Out[48]:
   start   end
0    5.0  40.0
1   60.0  95.0
```

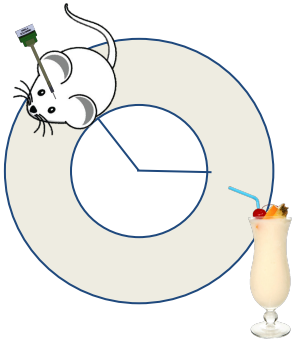
```
In [11]: lfp
Out[11]:
Time (s)
----------  ----------
0.0         -0.655732
0.1         -0.175004
0.2          0.55584
0.3          0.347774
0.4          0.0922895
...
99.5        -0.333844
99.6         0.145915
99.7        -0.377362
99.8        -0.466354
99.9         0.418279
dtype: float64, shape: (1000,)
```

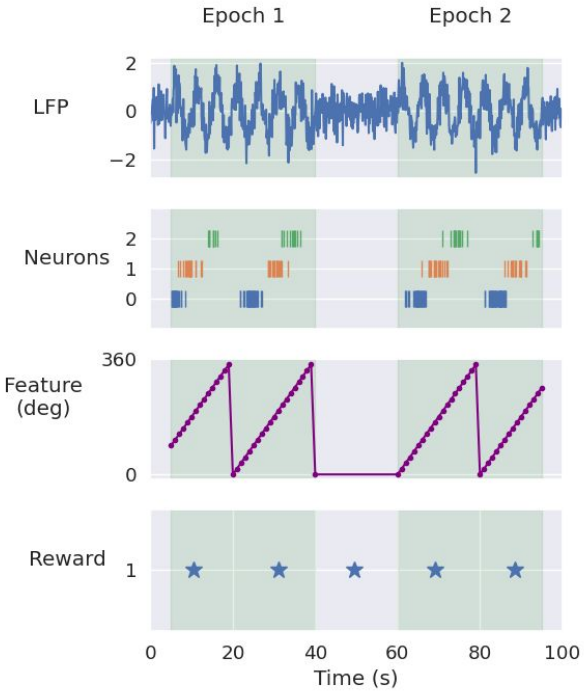


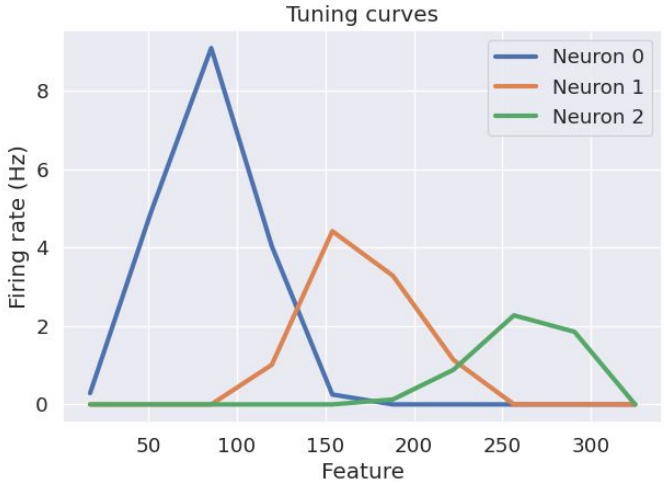

```
In [49]: spikes
Out[49]:
  Index    Freq. (Hz)
-------  ------------
      0          2.07
      1          1.08
      2          0.66
```

```
In [50]: feature
Out[50]:
Time (s)
0.0       0.0
1.0      18.0
2.0      36.0
3.0      54.0
4.0      72.0
        ...
95.0    270.0
96.0    288.0
97.0    306.0
98.0    324.0
99.0    342.0
Length: 100, dtype: float64
```

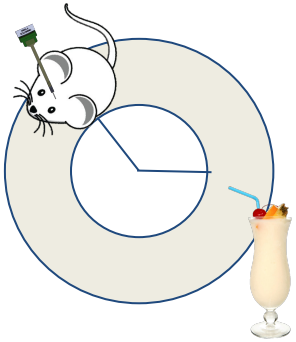
```
In [5]: reward
Out[5]:
Time (s)
10.841068764
31.582233204
51.683610725
71.239549326
91.661298984
shape: 5
```

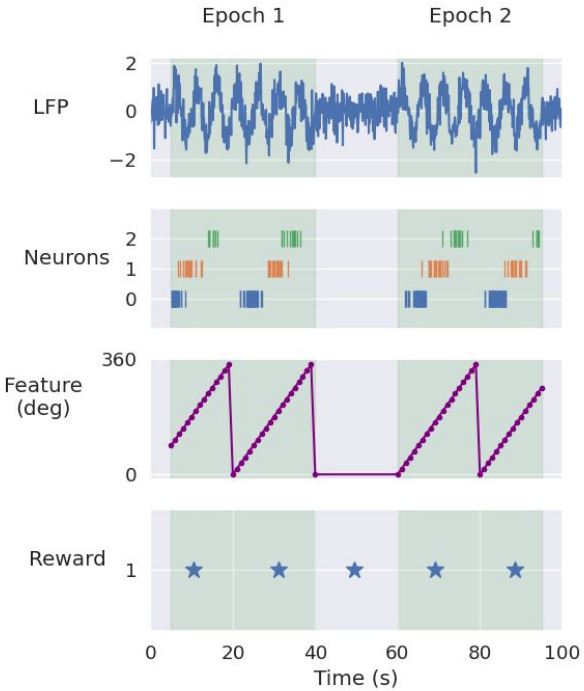






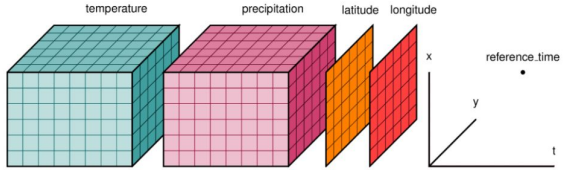

```python
tuning_curves = nap.compute_tuning_curves(
    data=spikes,
    features=feature,
    nb_bins=120,
    epochs=ep,
    range=(0, 360),
    feature_names=["feature"]
)
```



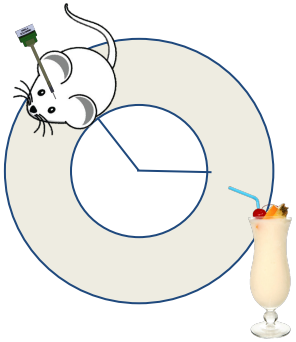


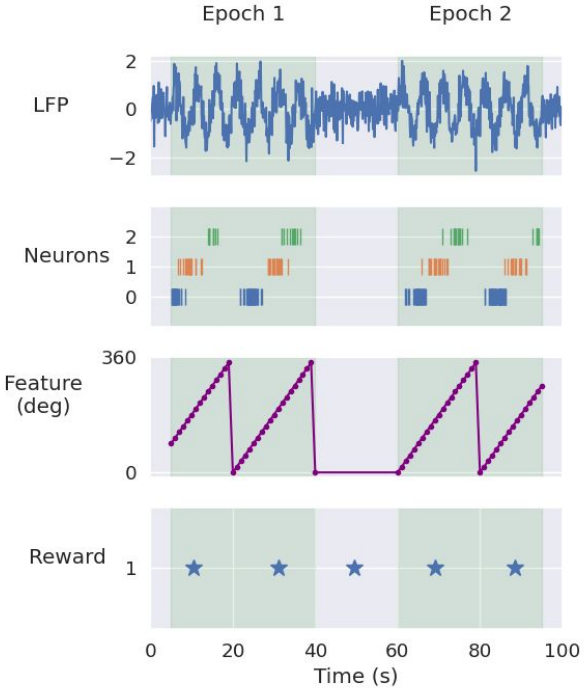
Xarray

N-D labeled arrays and datasets in Python




```python
tuning_curves = nap.compute_tuning_curves(
    data=spikes,
    features=feature,
    nb_bins=120,
    epochs=ep,
    range=(0, 360),
    feature_names=["feature"]
)
```

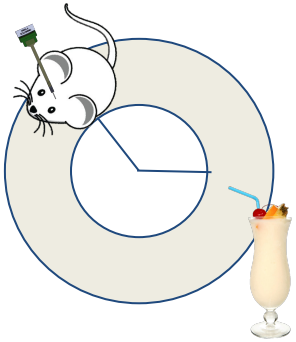


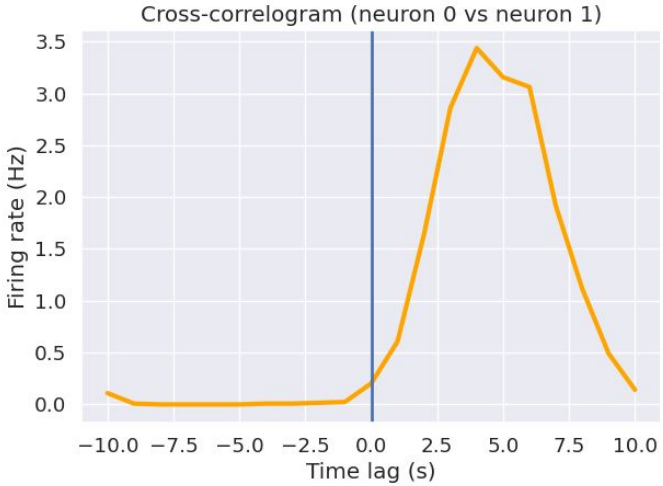


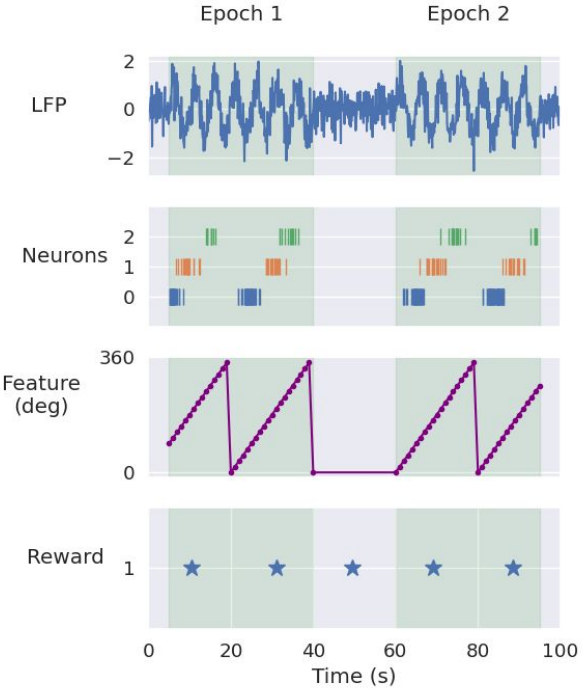
[6]:
```
print(tuning_curves)
```

```
<xarray.DataArray (unit: 3, feature: 12)> Size: 288B
array([[9.44444444e+00, 1.00000000e+01, 1.00000000e+01, 1.00000000e+01,
        1.00000000e+01, 1.00000000e+01, 1.00000000e+01, 1.00000000e+01,
        1.00000000e+01, 9.60000000e+00, 1.00000000e+01, 1.00000000e+01],
       [9.44444444e+01, 1.00000000e+02, 1.00000000e+02, 1.00000000e+02,
        1.00000000e+02, 1.00000000e+02, 1.00000000e+02, 1.00000000e+02,
        1.00000000e+02, 9.51000000e+01, 1.00000000e+02, 1.00000000e+02],
       [9.44444444e+03, 1.00000000e+04, 1.00000000e+04, 1.00000000e+04,
        1.00000000e+04, 1.00000000e+04, 1.00000000e+04, 1.00000000e+04,
        1.00000000e+04, 9.50010000e+03, 1.00000000e+04, 1.00000000e+04]])
Coordinates:
  * unit     (unit) int64 24B 0 1 2
  * feature  (feature) float64 96B 15.0 45.0 75.0 105.0 ... 285.0 315.0 345.0
Attributes:
    occupancy:  [ 9. 10.  6. 10.  8.  8.  8. 10.  6. 10.  8.  7.]
    bin_edges:  [array([  0.,  30.,  60.,  90., 120., 150., 180., 210., 240.,...
    fs:         1.0
```

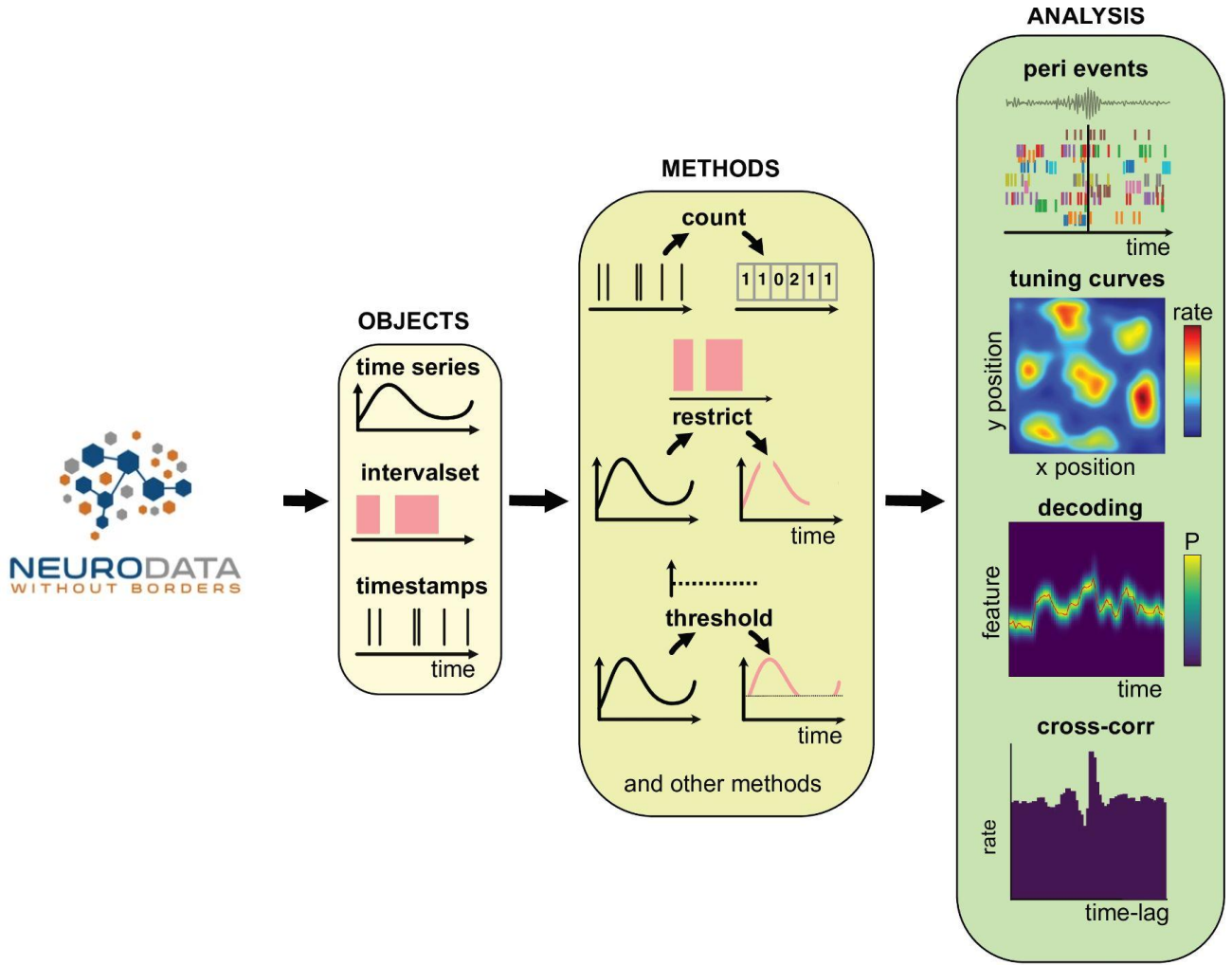
```
tuning_curves = nap.compute_tuning_curves(
    data=spikes,
    features=feature,
    nb_bins=120,
    epochs=ep,
    range=(0, 360),
    feature_names=["feature"]
)
```








```
cross_corr = nap.compute_crosscorrelogram(
    spikes,
    binsize = 1,
    windowsize = 10,
    ep = ep,
    norm = False)
```



FLATIRON INSTITUTE

Visualization

FLATIRON INSTITUTE

pynapple Public

PYthon Neural Analysis Package 🍍

Python ☆ 333 69

pynaviz Public

Python Neural Analysis Visualization

Python ☆ 8

pynapple Public

PYthon Neural Analysis Package 🍍

Python ☆ 333 69

pynaviz Public

Python Neural Analysis Visualization

Python ☆ 8

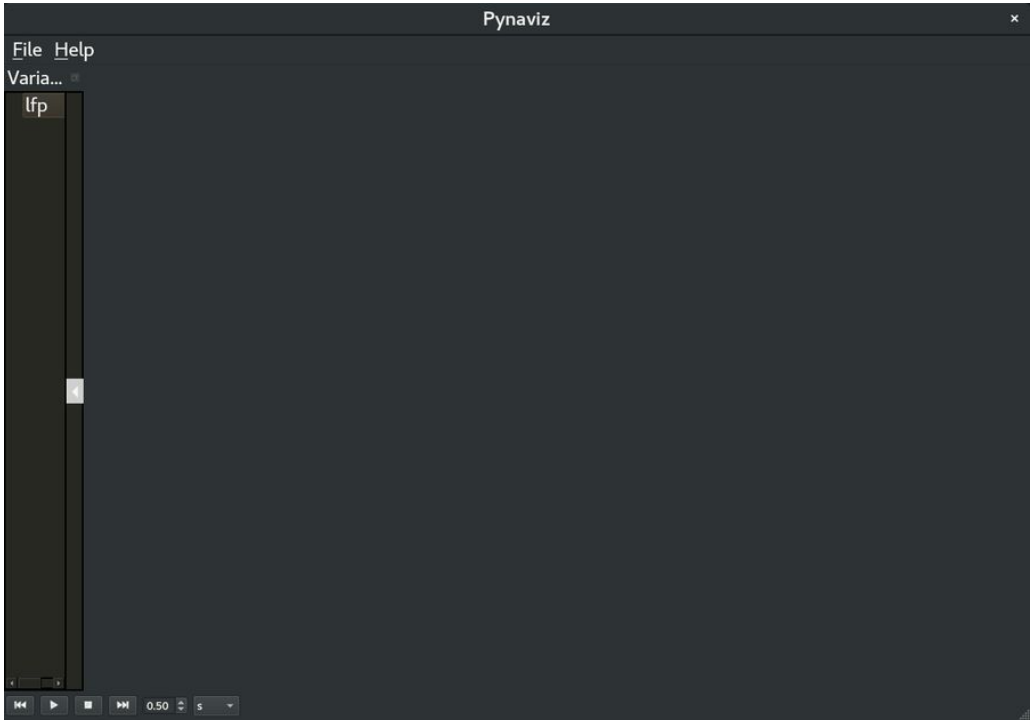
```python
import pynapple as nap
import numpy as np
from pynaviz import scope

# Create some example time series
tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

# Create a TsdFrame with metadata
tsdframe = nap.TsdFrame(
    t=np.arange(10000),
    d=np.random.randn(10000, 10),
    metadata={"label": np.random.randn(10)}
)

# Launch the visualization GUI
scope(globals())
```

Example Local Field Potential (LFP)

```python
import pynapple as nap
import numpy as np
from pynaviz import scope

path_to_dat_file = "lfp.dat"
fs = 20000 # Sampling frequency in Hz
num_channels = 16 # Number of channels
tsdframe = nap.misc.load_eeg(
    path_to_dat_file,
    n_channels=num_channels,
    frequency=fs
)
# Add channel metadata
tsdframe.channel = np.arange(0, num_channels)
# Add group metadata
tsdframe.group = np.hstack([[0]*10, [1]*6])

# Open the GUI
scope({"lfp":tsdframe})
```

Example Local Field Potential (LFP)

```python
import pynapple as nap
import numpy as np
from pynaviz import scope

path_to_dat_file = "lfp.dat"
fs = 20000 # Sampling frequency in Hz
num_channels = 16 # Number of channels
tsdframe = nap.misc.load_eeg(
    path_to_dat_file,
    n_channels=num_channels,
    frequency=fs
)
# Add channel metadata
tsdframe.channel = np.arange(0, num_channels)
# Add group metadata
tsdframe.group = np.hstack([[0]*10, [1]*6])

# Open the GUI
scope({"lfp":tsdframe})
```



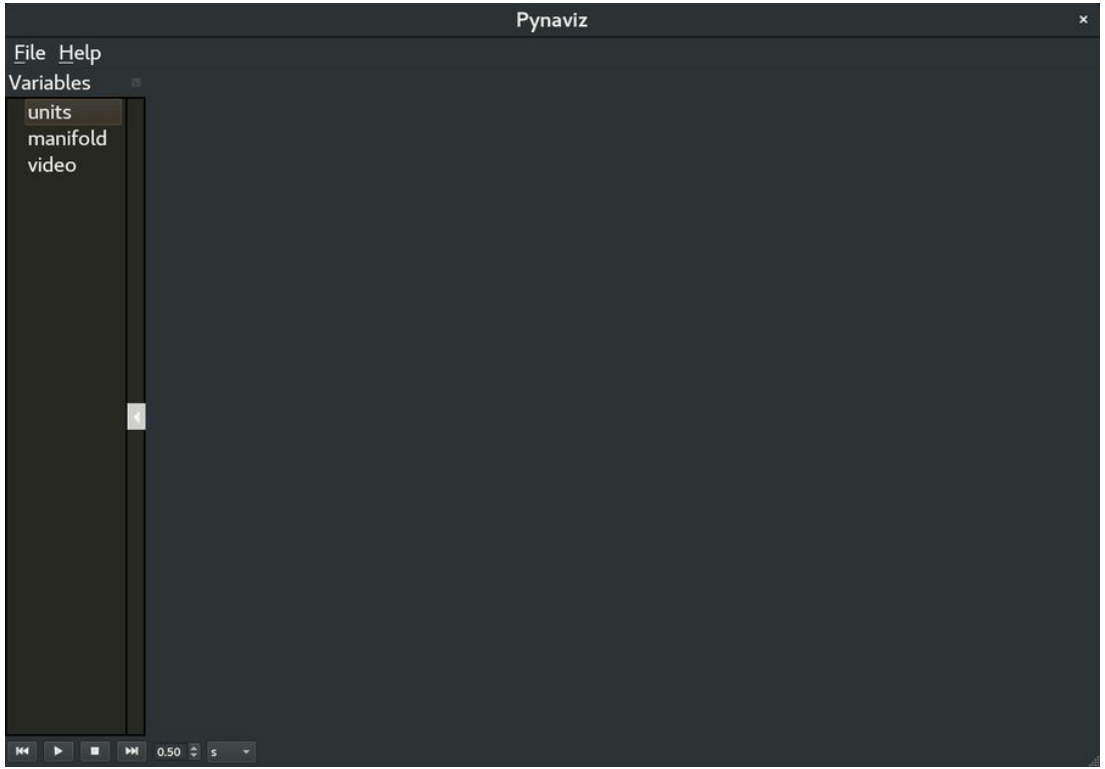
Spikes & 2-d manifold & video

FLATIRON INSTITUTE

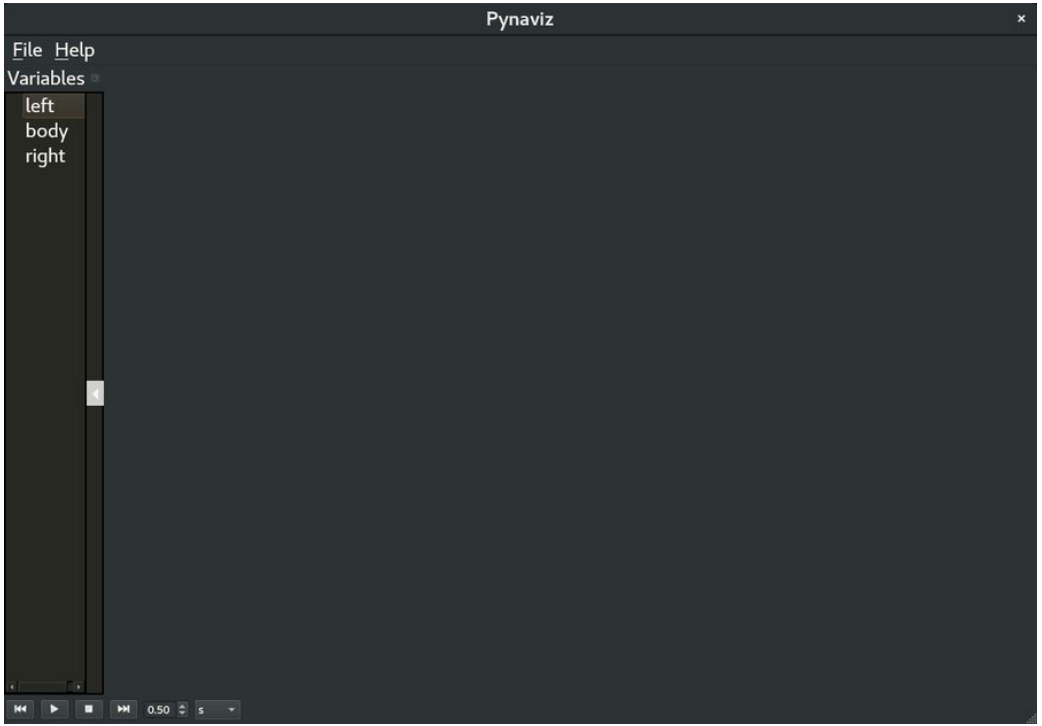
```
data = nap.load_file("A5044-240404A_wake.nwb")
units = data["units"]
manifold = data["manifold"]
video = viz.VideoWidget("A5044-240404A_wake.avi")

# Open the GUI
scope({
    "units": units,
    "manifold": manifold,
    "video": video}
)
```

Spikes & 2-d manifold & video

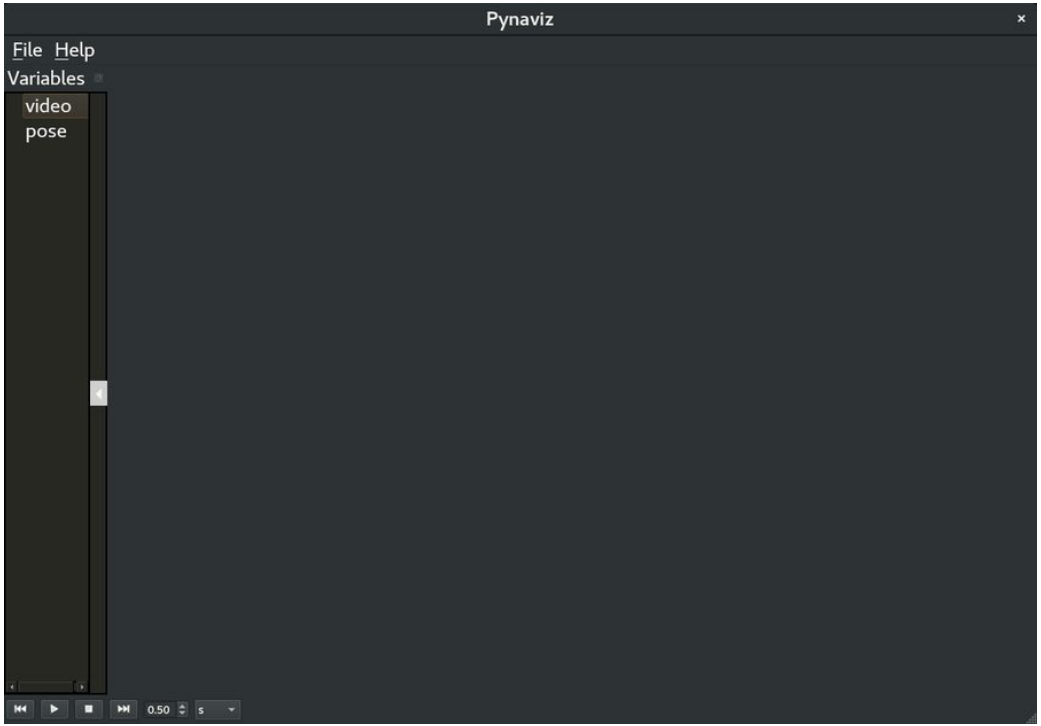
```
data = nap.load_file("A5044-240404A_wake.nwb")
units = data["units"]
manifold = data["manifold"]
video = viz.VideoWidget("A5044-240404A_wake.avi")

# Open the GUI
scope({
    "units": units,
    "manifold": manifold,
    "video": video}
)
```



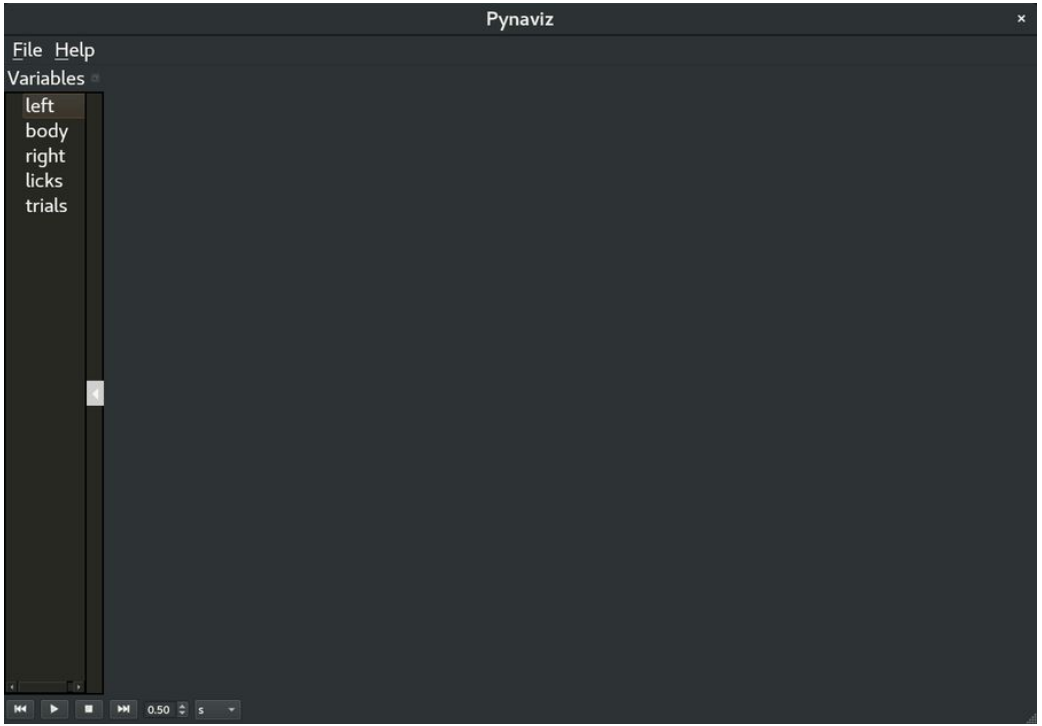
Multiple videos



Video & DeepLabCut



Video & Event & Trial intervals



Acceleration

```python
import pynapple as nap
import numpy as np

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

Run on CPU

```python
import pynapple as nap
import numpy as np

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

Run on CPU

```python
import pynapple as nap
import numpy as np

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

Run on CPU

$ pip install pynajax

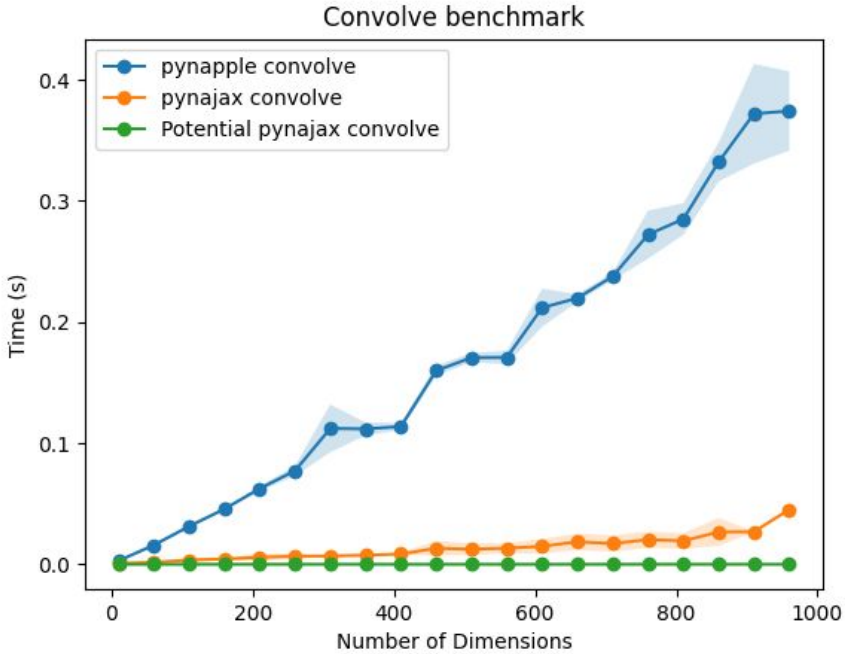
```python
import pynapple as nap
import numpy as np

nap.nap_config.set_backend("jax")

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

Run on GPU (potentially)

```python
import pynapple as nap
import numpy as np

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

$ pip install pynajax

```python
import pynapple as nap
import numpy as np

nap.nap_config.set_backend("jax")

tsd = nap.Tsd(t=np.arange(100), d=np.random.randn(100))

tsd.convolve(np.ones(11))
```

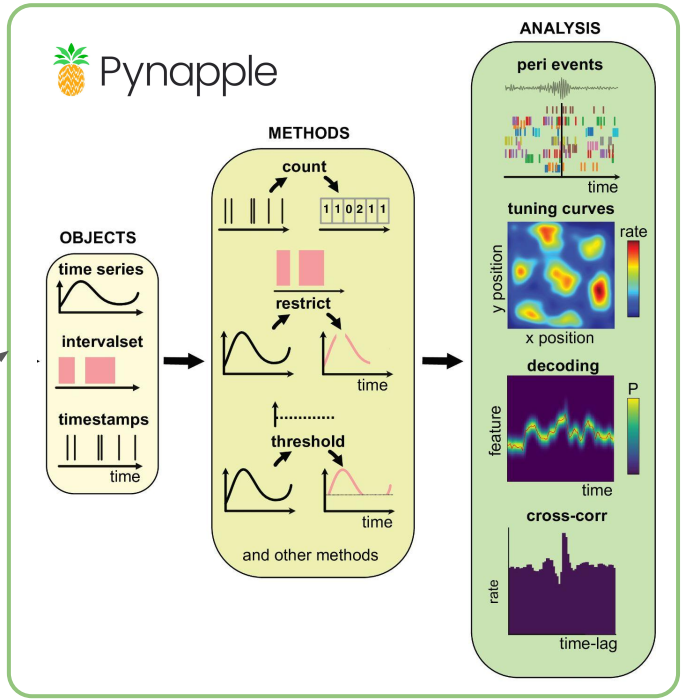


Time

Preprocessing
(CaImAn, SpikeInterface, …)

Pynapple

Postprocessing
(GLM, Manifold, …)



Pynapple

$ pip install pynapple

https://twitter.com/thepynapple

https://bsky.app/profile/pynapple.bsky.social

pynapple-org.slack.com

FLATIRON INSTITUTE

Pynapple

$ pip install pynapple

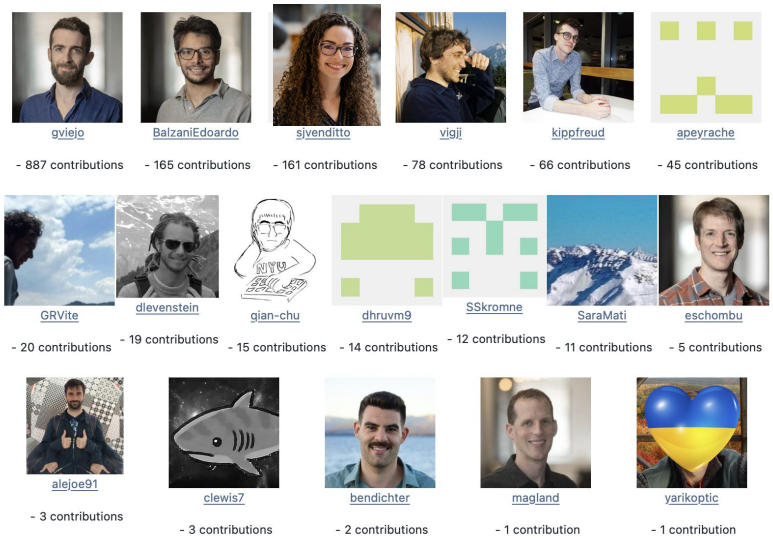
https://twitter.com/thepynapple

https://bsky.app/profile/pynapple.bsky.social

pynapple-org.slack.com

FLATIRON INSTITUTE

Contributors



Time for coding