

Nitime: an open-source package for time-series analysis of neuroscience data

<http://nipy.org/nitime>



NeuroImaging
in PYthon

Features:

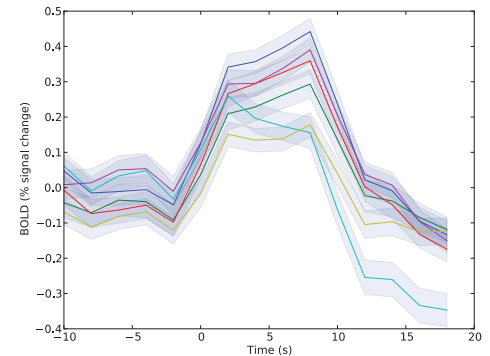
- >> Container objects for representation of time and for representation of time-series data:
- >> General purpose algorithm library (not dependent on the design of the time-series objects):
- >> 'Analyzer' objects bridge between the time-series representation and the algorithms, providing an easy-to-use interface
- >> Lazy initialization: intense computations in the analyzer objects are done on a need-to-know basis.
- >> Visualization of time-series data and results of analysis (see examples below).

Examples (see also: <http://nipy.org/nitime/examples/index.html>):

Event related analysis:

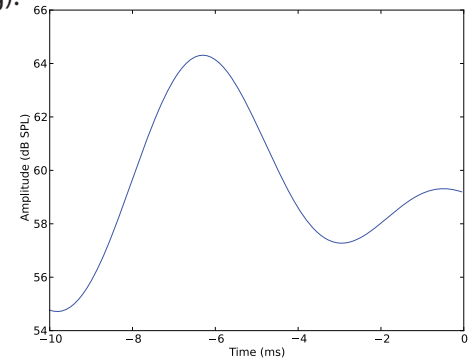
BOLD fMRI in human visual system in response to different directions of motion:

```
data = numpy.recfromcsv('data/event_related_fmri.csv')
# Initialize time-series objects:
T1 = nitime.TimeSeries(data.bold,sampling_interval=2)
T2 = nitime.TimeSeries(data.stimulus,sampling_interval=2)
# Initialize 'analyzer' object:
event_related = nitime.analysis.EventRelatedAnalyzer(T1,T2,15,offset=-5)
# Visualize result:
nitime.viz.plot_tseries(event_related.eta,ylabel='BOLD (% signal change)',
                        yerror=event_related.ets)
```



Intra-cellular recordings of spike-times in the grasshopper auditory system in response to an auditory stimulus (data available on the CRCNS data-sharing site: <http://crcns.org>):

```
# Load data:
stimulus = numpy.loadtxt('data/grasshopper_stimulus1.txt')
# Initialize time-series object:
T = nitime.TimeSeries(data=stimulus,sampling_interval=50,time_unit='us')
# Load the spike-times from the data file:
spike_times = numpy.loadtxt('data/grasshopper_spike_times1.txt')
# Initialize the Event object holding the spike-times:
E = nitime.Events(spike_times,time_unit='us')
# Initialize 'analyzer' object:
event_related = nitime.analysis.EventRelatedAnalyzer(T,E,200,offset=-200)
# Visualize the results:
nitime.viz.plot_tseries(event_related.eta,ylabel='Amplitude (dB SPL)',
                        time_unit='ms')
```



Coherence analysis:

Coherence is a spectral analogue of correlation: $R_{xy}(\lambda) = \frac{f_{xy}(\lambda)}{\sqrt{f_{xx}(\lambda) \cdot f_{yy}(\lambda)}}$

LFP data acquired with a polytrode in cat visual cortex (data available on CRCNS):

```
# Load the data:
data = [np.fromfile(f,dtype=np.uint16) for f in lfp_files]
# Initialize time-series
T = nitime.TimeSeries(data,time_unit='ms',sampling_interval=1)
# Initialize 'analyzer':
C = nitime.analysis.CoherenceAnalyzer(T)
# Get frequencies:
freqs = C.frequencies
# Low frequencies:
fig1=nitime.viz.drawmatrix_channels(np.mean(C.coherence[... ,np.where(freqs<50)[0]],-1))
fig1.axes[0].set_title('Low frequencies')
# High frequencies:
fig2=nitime.viz.drawmatrix_channels(np.mean(C.coherence[... ,np.where(freqs>70)[0]],-1))
fig2.axes[0].set_title('High frequencies')
```

