



nitime

Time-series analysis for fMRI data

Ariel Rokem

University of California, Berkeley

June 3rd, 2011



# Outline



# Outline



- Scientific motivation



# Outline



- Scientific motivation
- The nitime software library



# Outline



- Scientific motivation
- The nitime software library
- A simple example: coherence



# Outline



- Scientific motivation
- The nitime software library
- A simple example: coherence
- A scientifically interesting example



# Task-specific networks

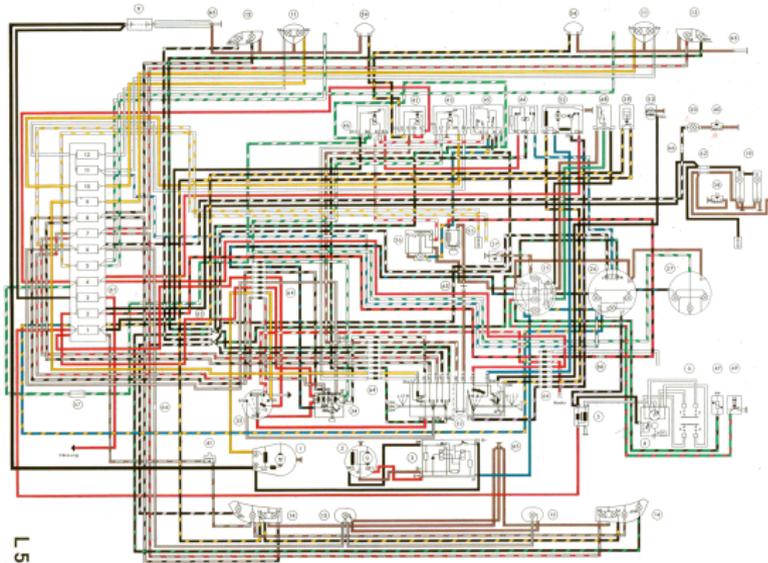


One of the goals of contemporary neuroscience is to delineate task-specific networks in the brain



# The wiring diagram

Wiring Diagram Type 912



Wiring Diagram Type 912

- ① stator
- ② generator
- ③ regulator
- ④ distributor
- ⑤ ignition coil
- ⑥ spark plugs
- ⑦ battery
- ⑧ headlight
- ⑨ parking and turn signal light
- ⑩ tail, stop, turn signal and back-up light
- ⑪ license plate light
- ⑫ interior light
- ⑬ luggage compartment light
- ⑭ large combination instrument
- ⑮ transistorized tachometer
- ⑯ speedometer
- ⑰ combined switch for turn signal, low beam, headlight signal, wiper, washer and horn button
- ⑱ ignition switch
- ⑲ main light switch
- ⑳ hand brake indicator light switch
- ㉑ door correct switch
- ㉒ stop light switch
- ㉓ luggage compartment light switch
- ㉔ back-up light switch
- ㉕ horn relay
- ㉖ hydraulic relay for high beam signal
- ㉗ turn signal flasher
- ㉘ warning signal flasher
- ㉙ warning signal relay
- ㉚ oil temperature sending unit
- ㉛ fuel pressure sending unit
- ㉜ oil pressure sending unit
- ㉝ water motor
- ㉞ washer pump
- ㉟ horn
- Ⓜ cigarette lighter
- Ⓝ warning signal switch
- Ⓞ fuse box
- Ⓟ two-pole plug connector
- Ⓠ cable distribution block
- Ⓡ right-way disconnect
- Ⓢ ground
- Ⓣ single-pole plug connector
- Ⓤ warning signal fuse with holder

**Fuses**

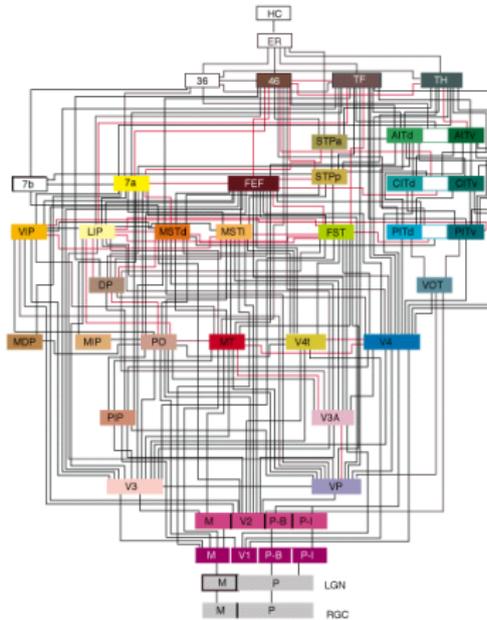
- 1 stop, turn signal and back-up light
- 2 interior light, cigarette lighter
- 3 supplementary heater (optional)
- 4 wiper and washer
- 5 fog lamps (optional)
- 6 license plate and luggage compartment
- 7 parking light, right
- 8 parking light, left
- 9 low beam, right
- 10 low beam, left
- 11 high beam, right, indicator light
- 12 high beam, left

L 53





# The wiring diagram



Felleman and Van Essen (1991)



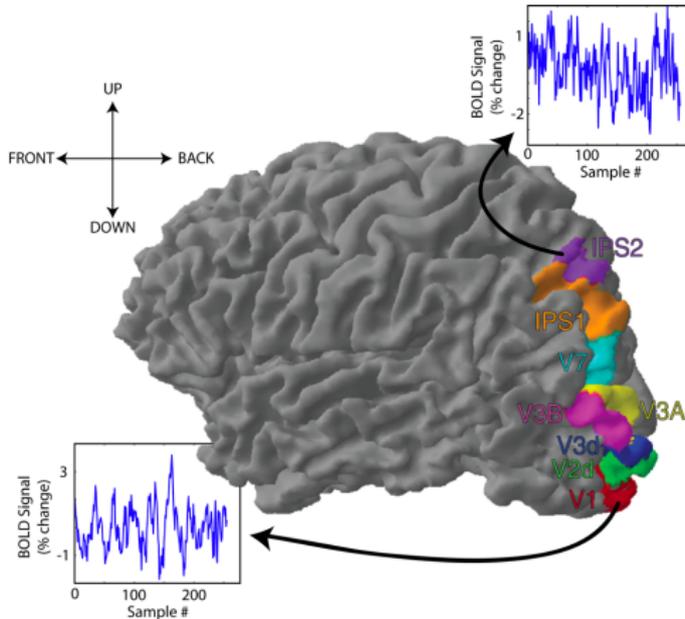
# Task-specific networks



One of the goals of contemporary neuroscience is to delineate task-specific networks in the brain



# Time-series in fMRI data



# Nitime



# Nitime



- Software library for the analysis of time-series from neuroscience data



# Nitime



- Software library for the analysis of time-series from neuroscience data
- Written in Python



# Nitime



- Software library for the analysis of time-series from neuroscience data
- Written in Python
- Free and open source



# Nitime



- Software library for the analysis of time-series from neuroscience data
- Written in Python
- Free and open source
- Part of the NIPY project



# Nitime



- Software library for the analysis of time-series from neuroscience data
- Written in Python
- Free and open source
- Part of the NIPY project
- <http://nipy.org/nitime>



nitime components:



# nitime components:



- `nitime.timeseries`



# nitime components:

- `nitime.timeseries`
- `nitime.viz`



# nitime components:

- `nitime.timeseries`
- `nitime.viz`
- `nitime.algorithms`



# nitime components:

- `nitime.timeseries`
- `nitime.viz`
- `nitime.algorithms`
- `nitime.analysis`



# nitime components:

- `nitime.timeseries`
- `nitime.viz`
- `nitime.algorithms`
- `nitime.analysis`
- `nitime.utils`



# TimeSeries example:



## TimeSeries example:



```
>>> import nitime.timeseries as ts
>>> t1 = ts.TimeSeries(data=[1,2,3,4],
                       sampling_interval=1.5,
                       time_unit='s')
```



## TimeSeries example:

```
>>> import nitime.timeseries as ts
>>> t1 = ts.TimeSeries(data=[1,2,3,4],
                        sampling_interval=1.5,
                        time_unit='s')

>>> t1.time
UniformTime([ 0. ,  1.5,  3. ,  4.5],
            time_unit='s')
```



## TimeSeries example:

```
>>> import nitime.timeseries as ts
>>> t1 = ts.TimeSeries(data=[1,2,3,4],
                        sampling_interval=1.5,
                        time_unit='s')

>>> t1.time
UniformTime([ 0. ,  1.5,  3. ,  4.5],
             time_unit='s')

>>> t1.sampling_rate
0.666666666667 Hz
```



# Vizualization



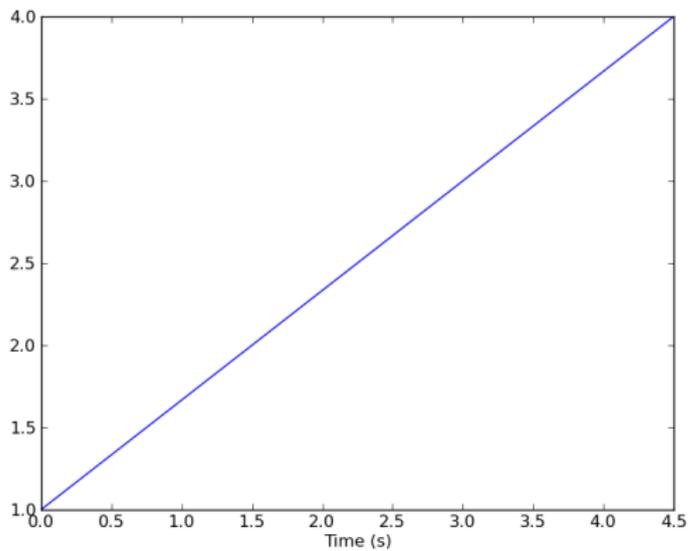
# Vizualization



```
>>> import nitime.viz as viz  
>>> viz.plot_tseries(t1)
```



# Vizualization



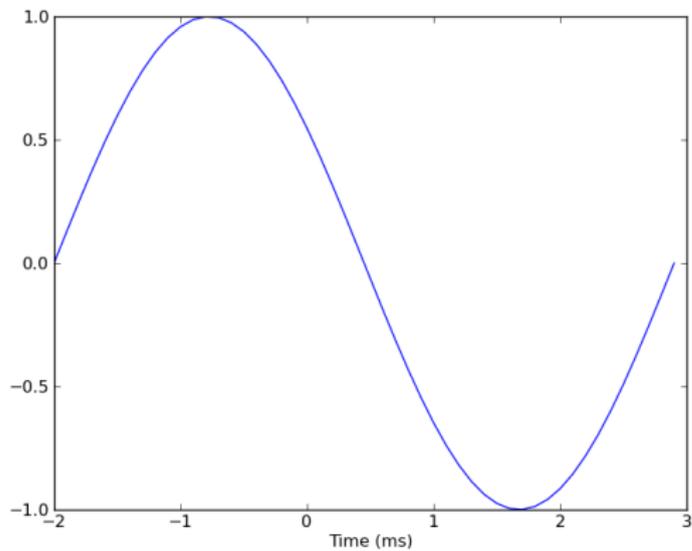
# Vizualization



```
>>> t2 = ts.TimeSeries(data=np.sin(  
                        np.linspace(0,2*np.pi)),  
                        sampling_interval=0.1,  
                        t0=-2,  
                        time_unit='ms')  
  
>>> viz.plot_tseries(t2)
```



# Vizualization



nitime.algorithms



# nitime.algorithms



- Provides a functional interface based on numpy arrays



# nitime.algorithms



- Provides a functional interface based on numpy arrays
- coherence



# nitime.algorithms



- Provides a functional interface based on numpy arrays
- coherence
- event-related



# nitime.algorithms



- Provides a functional interface based on numpy arrays
- coherence
- event-related
- filter



# nitime.algorithms



- Provides a functional interface based on numpy arrays
- coherence
- event-related
- filter
- spectral



# nitime.algorithms



- Provides a functional interface based on numpy arrays
- coherence
- event-related
- filter
- spectral
- autoregressive





- Provides a functional interface based on numpy arrays
- coherence
- event-related
- filter
- spectral
- autoregressive
- These implementations accept arrays as their inputs, not TimeSeries!



# Example: univariate analysis



## Example: univariate analysis

```
>>> noise = 0.5
>>> t = np.linspace(0,8*pi,1024)

>>> x = (np.sin(5*t) + np.sin(1.33*t) +
        noise*np.random.randn(t.shape[-1]))

>>> y = (np.sin(5*t + pi/4) +
        np.sin(1.33*t-pi/2) +
        noise*np.random.randn(t.shape[-1]))
```



## Example: univariate analysis

```
>>> noise = 0.5
>>> t = np.linspace(0,8*pi,1024)

>>> x = (np.sin(5*t) + np.sin(1.33*t) +
        noise*np.random.randn(t.shape[-1]))

>>> y = (np.sin(5*t + pi/4) +
        np.sin(1.33*t-pi/2) +
        noise*np.random.randn(t.shape[-1]))

>>> t3 = ts.TimeSeries(np.vstack([x,y]),
                       sampling_rate=np.pi)
```



## Example: univariate analysis

```
>>> noise = 0.5
>>> t = np.linspace(0,8*pi,1024)

>>> x = (np.sin(5*t) + np.sin(1.33*t) +
        noise*np.random.randn(t.shape[-1]))

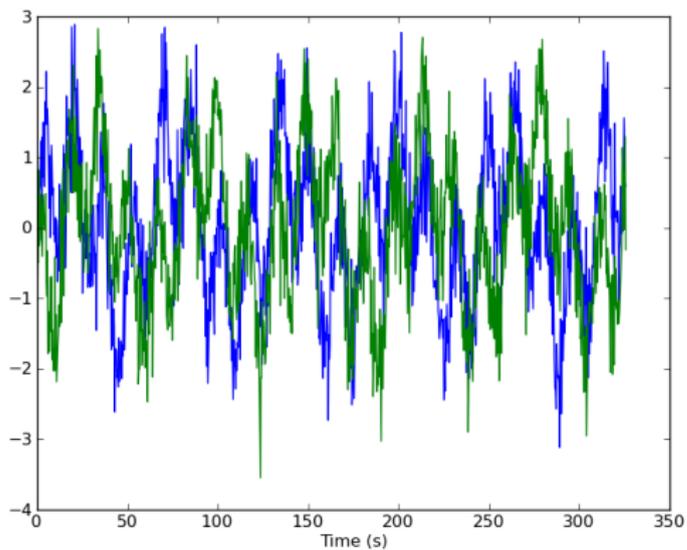
>>> y = (np.sin(5*t + pi/4) +
        np.sin(1.33*t-pi/2) +
        noise*np.random.randn(t.shape[-1]))

>>> t3 = ts.TimeSeries(np.vstack([x,y]),
                      sampling_rate=np.pi)

>>> viz.plot_tseries(t3)
```



# Time-series



# Univariate analysis: spectral analysis



```
>>> import nitime.algorithms as alg
```



# Univariate analysis: spectral analysis



```
>>> import nitime.algorithms as alg

>>> method = {'this_method': 'welch',
              'Fs': np.pi,
              'NFFT': 256}
```



# Univariate analysis: spectral analysis



```
>>> import nitime.algorithms as alg
```

```
>>> method = {'this_method': 'welch',  
              'Fs': np.pi,  
              'NFFT': 256}
```

```
>>> f, c = alg.get_spectra(t3.data, method=method)
```



# Univariate analysis: spectral analysis



```
>>> import nitime.algorithms as alg

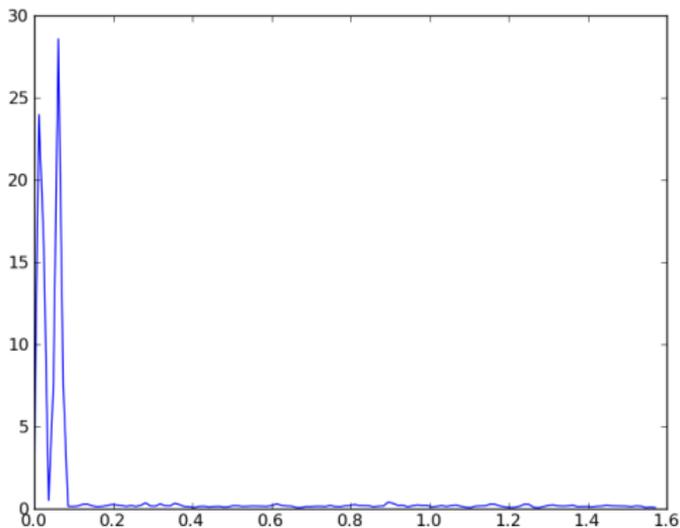
>>> method = {'this_method': 'welch',
              'Fs': np.pi,
              'NFFT': 256}

>>> f, c = alg.get_spectra(t3.data, method=method)

>>> import matplotlib.pyplot as plt
>>> plt.plot(f, c[0,0])
```



# Univariate analysis: spectral analysis



nitime.analysis



# nitime.analysis

- Provides a more stream-lined analysis interface



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence
- correlation



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence
- correlation
- event-related



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence
- correlation
- event-related
- normalization



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence
- correlation
- event-related
- normalization
- snr



# nitime.analysis



- Provides a more stream-lined analysis interface
- 'Knows' about TimeSeries objects and accepts them as inputs
- coherence
- correlation
- event-related
- normalization
- snr
- spectral



# Bivariate analysis 1: cross-correlation



$$R_{xy}(\tau) = \sum_{i=0}^T x(t)y(t + \tau)$$



# Bivariate analysis 1: cross-correlation



# Bivariate analysis 1: cross-correlation



```
>>> import nitime.analysis as nta
```



# Bivariate analysis 1: cross-correlation



```
>>> import nitime.analysis as nta
```

```
>>> XC = nta.CorrelationAnalyzer(t3)
```



# Bivariate analysis 1: cross-correlation



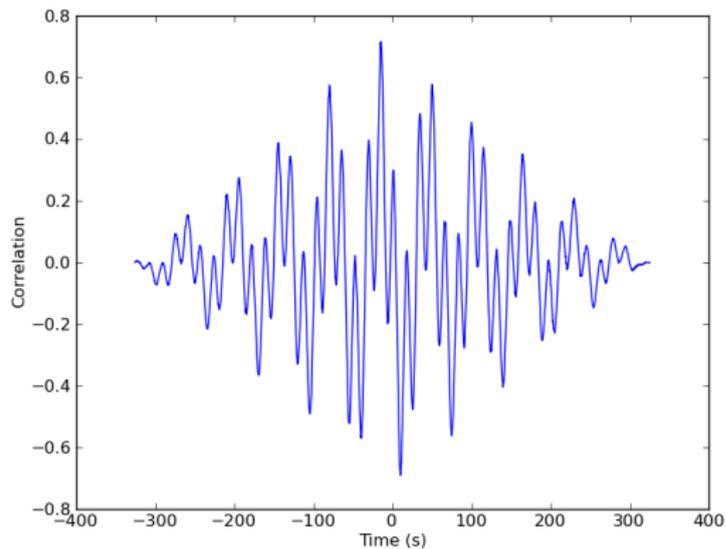
```
>>> import nitime.analysis as nta
```

```
>>> XC = nta.CorrelationAnalyzer(t3)
```

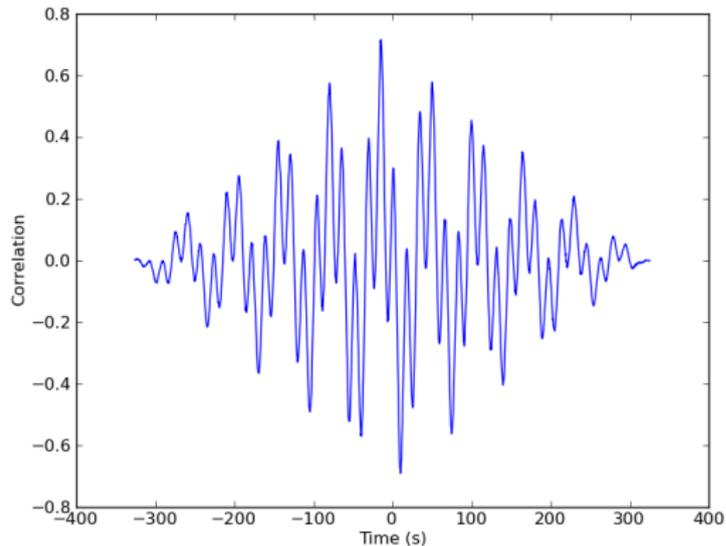
```
>>> viz.plot_xcorr(XC.xcorr_norm,[[0,1]])
```



# Bivariate analysis 1: cross-correlation



# Bivariate analysis 1: cross-correlation



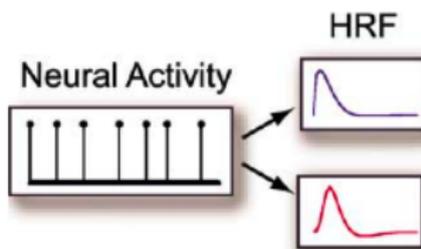
$$r_{xy} = 0.25$$



# Why should we care?

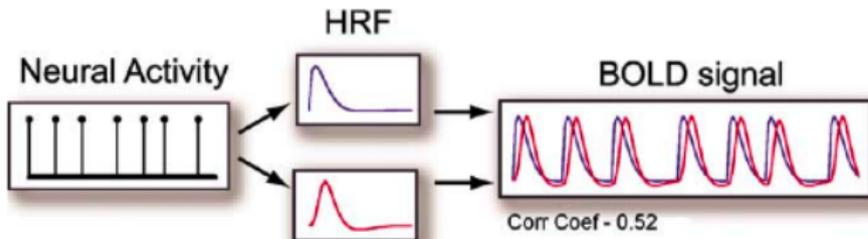


Hemodynamic delays



# Why should we care?

## Hemodynamic delays



# Coherency



A spectral analog of correlation



# Coherency



$$C_{xy}(\omega) = \frac{f_{xy}(\omega)}{\sqrt{f_{xx}(\omega)f_{yy}(\omega)}}$$



# Coherency

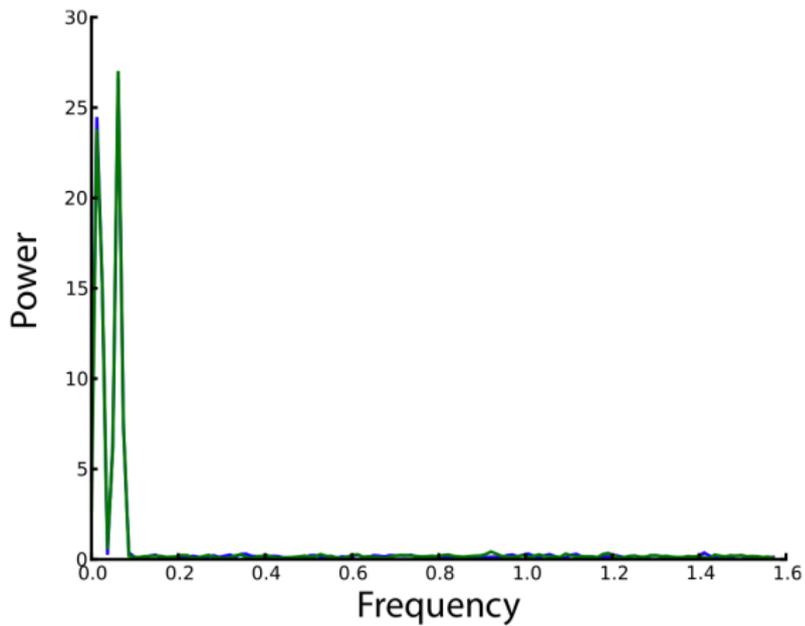


$$C_{xy}(\omega) = \frac{f_{xy}(\omega)}{\sqrt{f_{xx}(\omega)f_{yy}(\omega)}}$$

Where,  $f_{xy}$  is the cross-spectral density and  $f_{xx}$  is the PSD of  $x$



# Coherency



# Coherency



Coherency is complex-valued:



# Coherency



Coherency is complex-valued:

Coherence:  $Coh_{xy}(\omega) = abs(C_{xy}(\omega)) =$



# Coherency



Coherency is complex-valued:

$$\text{Coherence: } \text{Coh}_{xy}(\omega) = \text{abs}(C_{xy}(\omega)) = \frac{|f_{xy}(\omega)|^2}{f_{xx}(\omega)f_{yy}(\omega)}$$



# Coherency



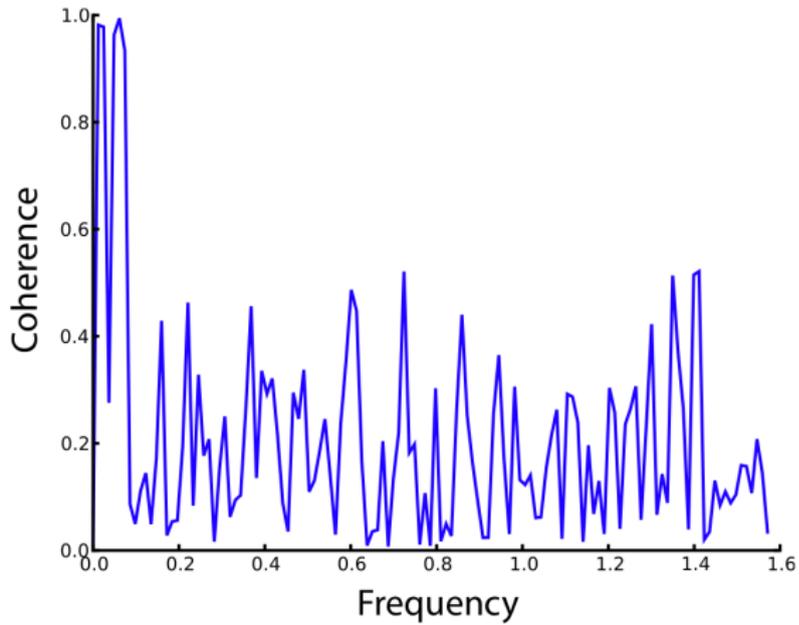
Coherency is complex-valued:

$$\text{Coherence: } \text{Coh}_{xy}(\omega) = \text{abs}(C_{xy}(\omega)) = \frac{|f_{xy}(\omega)|^2}{f_{xx}(\omega)f_{yy}(\omega)}$$

Ranges from 0 to 1



# Coherence



# Coherency



Coherency is complex-valued:

$$\text{Coherence: } \text{Coh}_{xy}(\omega) = \text{abs}(C_{xy}(\omega)) = \frac{|f_{xy}(\omega)|^2}{f_{xx}(\omega)f_{yy}(\omega)}$$



# Coherency



Coherency is complex-valued:

$$\text{Coherence: } \text{Coh}_{xy}(\omega) = \text{abs}(C_{xy}(\omega)) = \frac{|f_{xy}(\omega)|^2}{f_{xx}(\omega)f_{yy}(\omega)}$$

$$\text{Phase: } \phi(\omega) = \text{angle}(C_{xy}) = \tan^{-1} \frac{\Im(f_{xy}(\omega))}{\Re(f_{xy}(\omega))}$$



# Coherency



Coherency is complex-valued:

$$\text{Coherence: } \text{Coh}_{xy}(\omega) = \text{abs}(C_{xy}(\omega)) = \frac{|f_{xy}(\omega)|^2}{f_{xx}(\omega)f_{yy}(\omega)}$$

$$\text{Phase: } \phi(\omega) = \text{angle}(C_{xy}) = \tan^{-1} \frac{\Im(f_{xy}(\omega))}{\Re(f_{xy}(\omega))}$$

Ranges from  $-\pi$  to  $\pi$



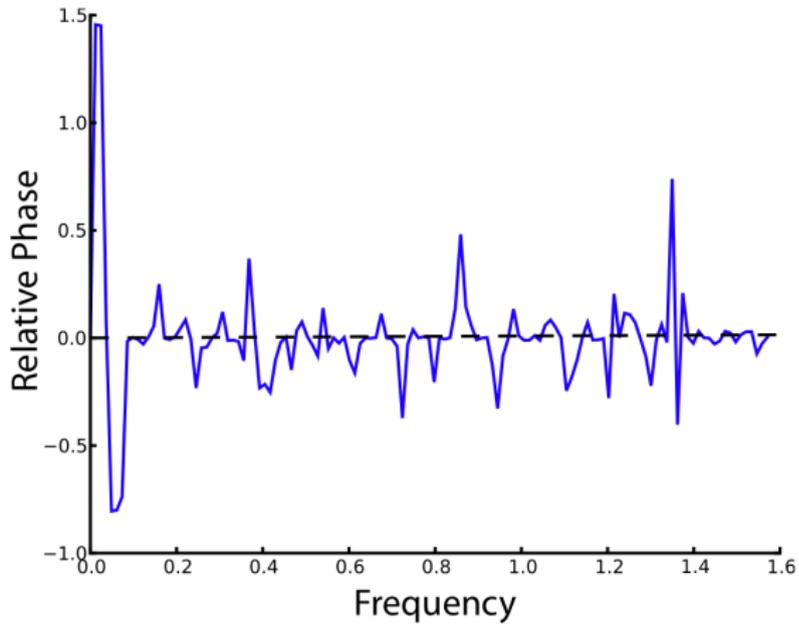
# Phase



```
>>> plt.plot(C.frequencies, C.phase[0,1])
```



# Phase



# Not surprising



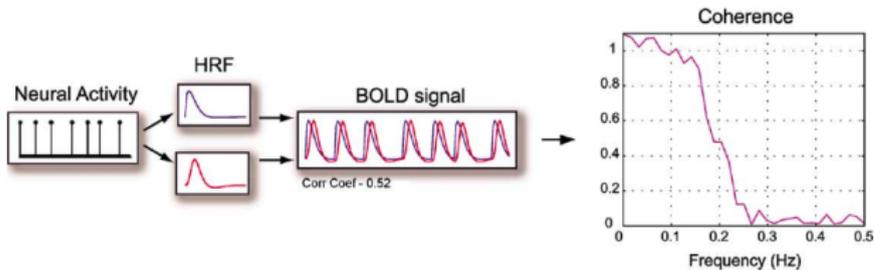
```
>>> noise = 0.5
>>> t = np.linspace(0,8*pi,1024)

>>> x = (np.sin(5*t) + np.sin(1.33*t) +
        noise*np.random.randn(t.shape[-1]))

>>> y = (np.sin(5*t + pi/4) +
        np.sin(1.33*t-pi/2) +
        noise*np.random.randn(t.shape[-1]))
```



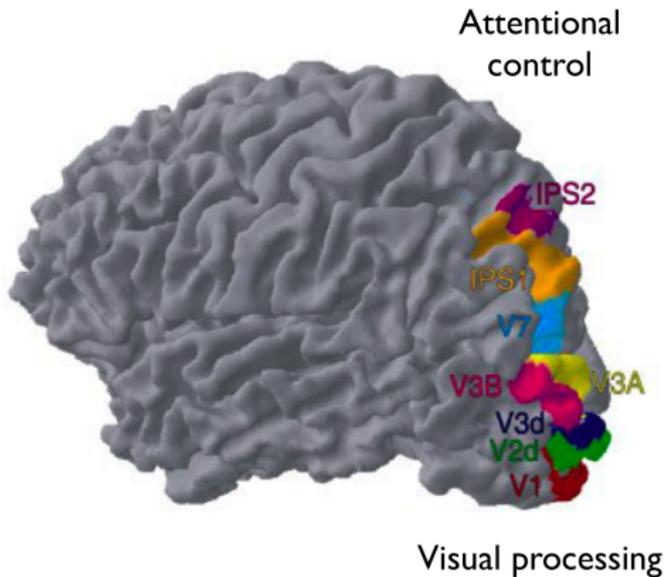
# Coherency



Sun et al. (2004,2005)



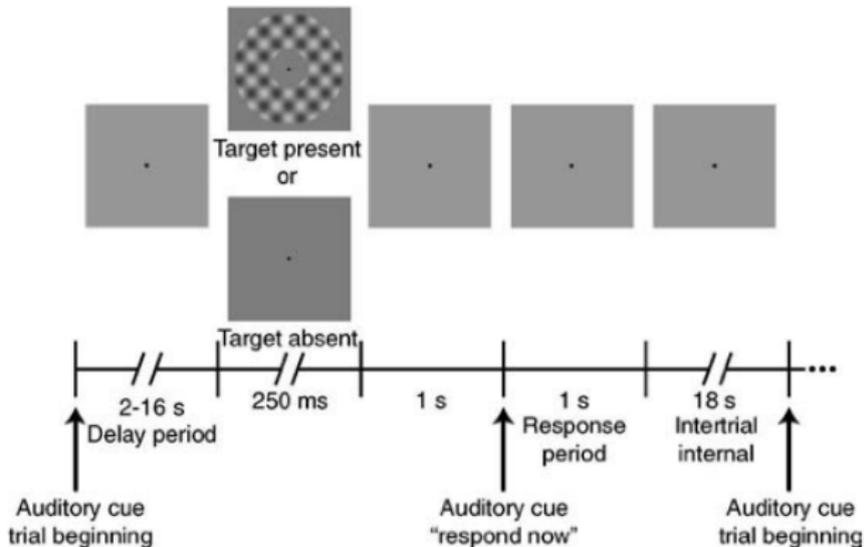
# Using coherency to study task-related networks



Lauritzen et al. (2009)



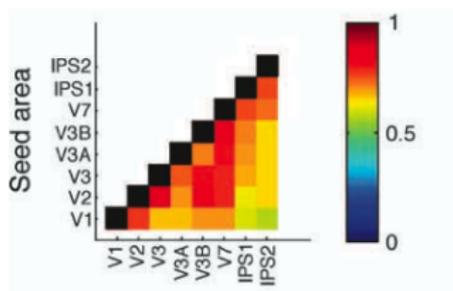
# Covert attention task



Lauritzen et al. (2009)



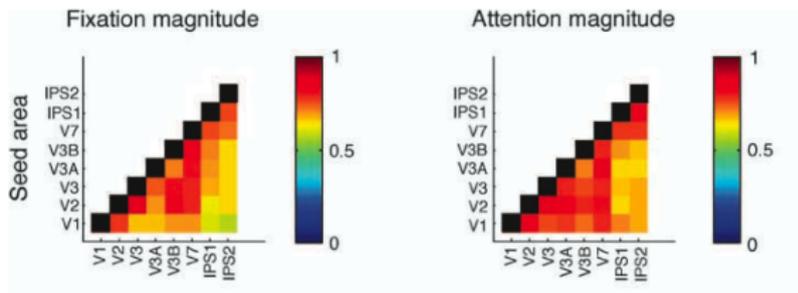
# Baseline coherence



Lauritzen et al. (2009)



# Attention coherence



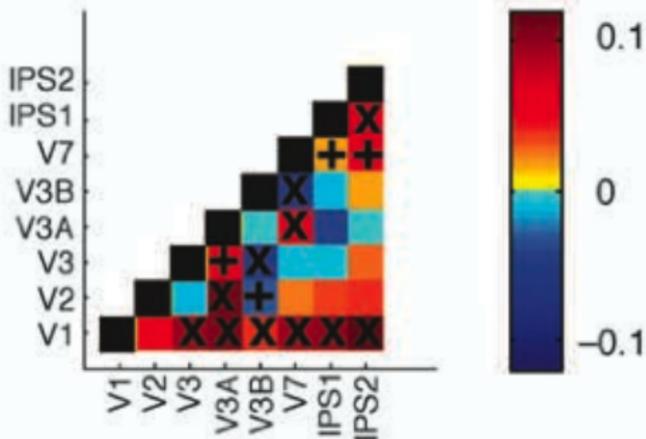
Lauritzen et al. (2009)



# Difference in coherence



## C Attention magnitude- fixation magnitude



Lauritzen et al. (2009)



# Time delay



The time-delay between two time-series can be calculated from the phase delay



# Time delay

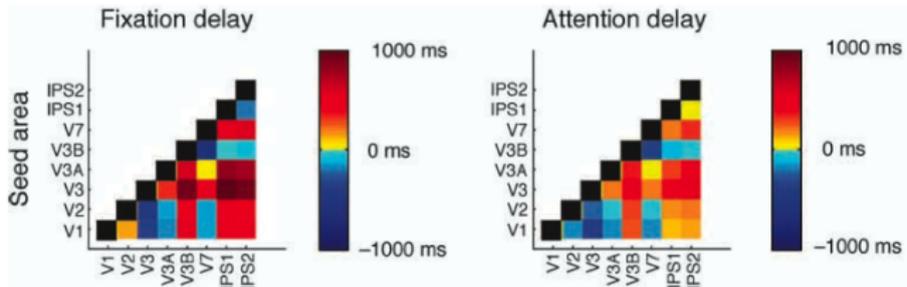


The time-delay between two time-series can be calculated from the phase delay

$$\Delta t(\omega) = \frac{\phi(\omega)}{2\pi\omega}$$



# Time delay



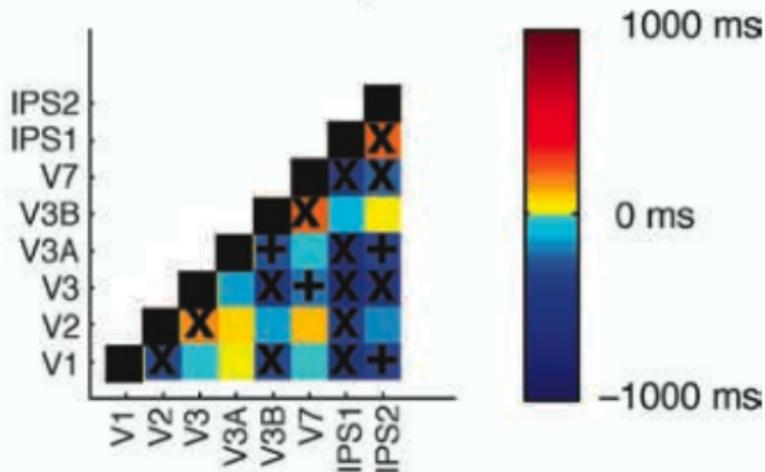
Lauritzen et al. (2009)



# Delay difference



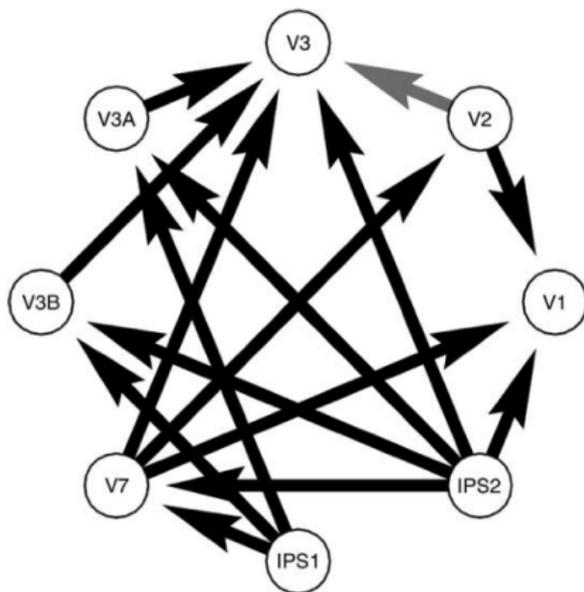
F Attention delay-  
fixation delay



Lauritzen et al. (2009)



# Task-related network changes:



Lauritzen et al. (2009)



# Coherency analysis: summary



# Coherency analysis: summary



- Coherency analysis is a spectral analog of correlation



# Coherency analysis: summary



- Coherency analysis is a spectral analog of correlation
- Overcomes some of the pitfalls of 0-order correlation



# Coherency analysis: summary



- Coherency analysis is a spectral analog of correlation
- Overcomes some of the pitfalls of 0-order correlation
- But - when used for fMRI need to pay attention to:



# Coherency analysis: summary



- Coherency analysis is a spectral analog of correlation
- Overcomes some of the pitfalls of 0-order correlation
- But - when used for fMRI need to pay attention to:
- Baseline connectivity



# Coherency analysis: summary



- Coherency analysis is a spectral analog of correlation
- Overcomes some of the pitfalls of 0-order correlation
- But - when used for fMRI need to pay attention to:
- Baseline connectivity
- Confounds due to HRF differences



# More examples



<http://nipy.org/nitime/examples>



# Summary: nitime



# Summary: nitime



- nitime provides tools for representing and visualizing time-series and derived quantities



# Summary: nitime



- nitime provides tools for representing and visualizing time-series and derived quantities
- Implements several univariate and bi-/multi-variate algorithms for time-series analysis



# Summary: nitime



- nitime provides tools for representing and visualizing time-series and derived quantities
- Implements several univariate and bi-/multi-variate algorithms for time-series analysis
- Allows for flexible use: choose the `analysis` module if you prefer a more stream-lined interface to the algorithms



# Summary: nitime

- nitime provides tools for representing and visualizing time-series and derived quantities
- Implements several univariate and bi-/multi-variate algorithms for time-series analysis
- Allows for flexible use: choose the `analysis` module if you prefer a more stream-lined interface to the algorithms
- Or the `algorithms` module if you prefer more customability



Stay tuned



# Stay tuned

- 0.3 release in the next couple of weeks



# Stay tuned



- 0.3 release in the next couple of weeks
- Which will include new stuff:



# Stay tuned



- 0.3 release in the next couple of weeks
- Which will include new stuff:
- Granger causality



# Stay tuned



- 0.3 release in the next couple of weeks
- Which will include new stuff:
- Granger causality
- Future work:



# Stay tuned



- 0.3 release in the next couple of weeks
- Which will include new stuff:
- Granger causality
- Future work:
- Network analysis



Thanks!



# Thanks!

Fernando Perez  
Mike Trumpis  
Kilian Koepsell  
Paul Ivanov



# Thanks!



Fernando Perez  
Mike Trumpis  
Kilian Koepsell  
Paul Ivanov  
The NIPY developers  
Matthew Brett



# Thanks!



Fernando Perez

Mike Trumpis

Kilian Koepsell

Paul Ivanov

The NIPY developers

Matthew Brett

Neurodebian (Yaroslav and Michael)



# Thanks!



Fernando Perez

Mike Trumpis

Kilian Koepsell

Paul Ivanov

The NIPY developers

Matthew Brett

Neurodebian (Yaroslav and Michael)

Emi Nomura

Caterina Gratton

Ayelet Landau

Thomas Lauritzen



# Thanks!



Fernando Perez

Mike Trumpis

Kilian Koepsell

Paul Ivanov

The NIPY developers

Matthew Brett

Neurodebian (Yaroslav and Michael)

Emi Nomura

Caterina Gratton

Ayelet Landau

Thomas Lauritzen

Mark D'Esposito

Michael Silver

