neo

Samuel Garcia

neo : a 100% code jam project!

History :

- Discussions started in Freiburg 2009 (codejam #3) : Pierre Y., Andrew D, Luc E., …
- Neo 0.1 release some mouth. Coded almost alone.
- Neo 0.1 presented in Marseille 2010 (codejam #4)
- And New discussion for neo 0.2 : Andrey S, Philipp R, Andrew D, Florent J, …
- Private code jam with new team in Gif at Andrew's lab last year.
- Released on feb 2012. neo 0.2
- Presentation Edinburgh 2012 (codemjam #5)

What is neo ?

neo.core = a simple and intuitive set on objects for representing electrophysiological dataset in python.

neo.io = a common layer for reading/writing in the cacophony of file formats.

Goals ?

What are main interests :
• Interoperability between projects (g-node, pynn, OpenElectrophy, NeuroTools, ...)
• A 5 min. installable, multiplatfrom, and easy to play file reader.

Dependencies ?

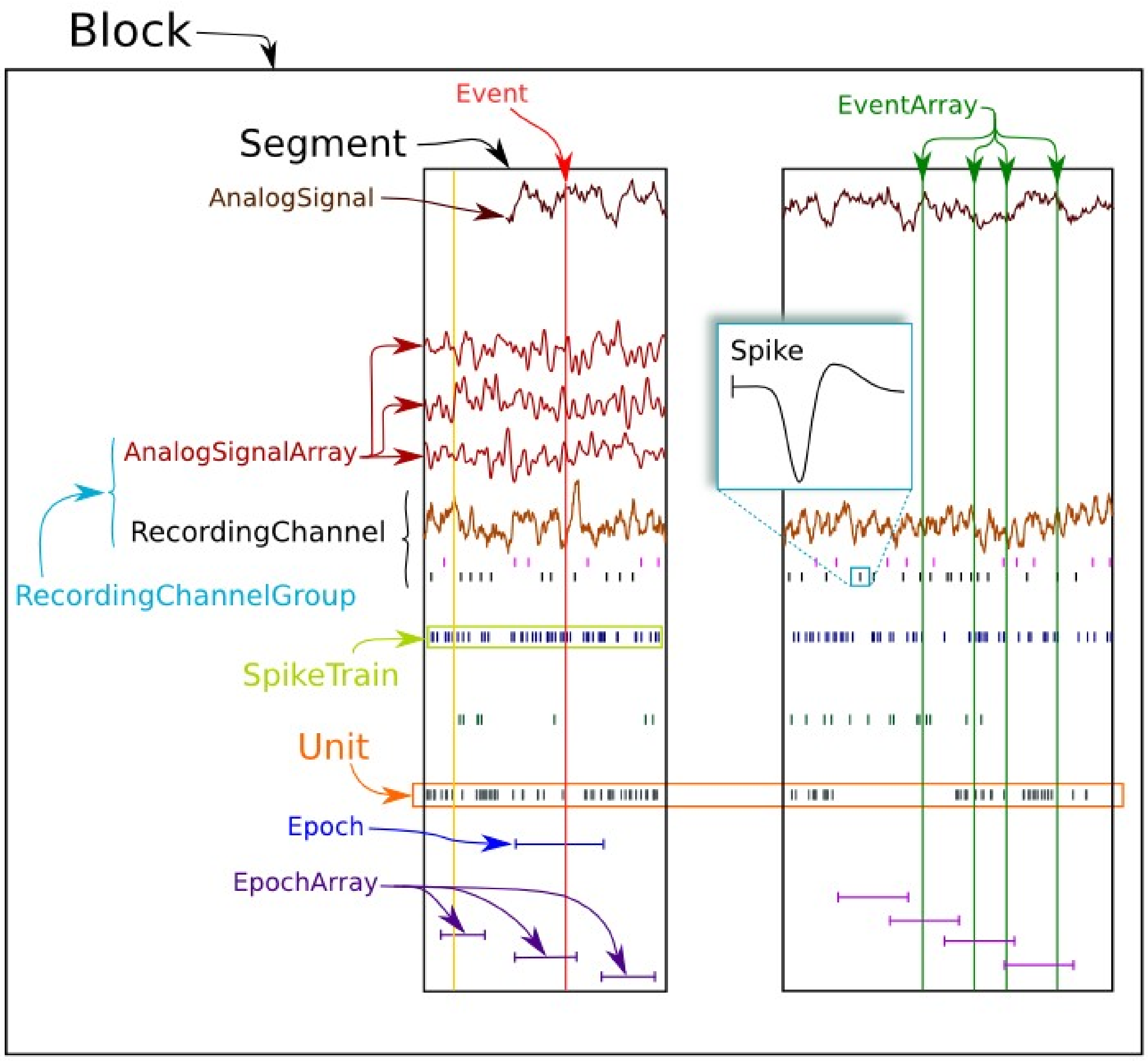
Few = numpy and quantities
Optional for some IOs = pytables , scipy,

Equivalent project
• Neuroshare (ddl provide commercial)
• for neuro imaging: nibabel (python)

What is new ?

- new schema more consistent.
- new objects
- New IOs
- use the quantities module for everything that can have units.
- Python 3 support
- better tests
- Doc with better English grammar.

Class tour



Neo 0.2 architecture

Class tour: Concept

3 types of objects:
- Data objects : AnalogSignal, SpikeTrain, EventArray, EpochArray
- Containers objects : Block, Segment
- Grouping objects : RecordingChannel, RecordingChannelGroup, Unit (ex Neuron)

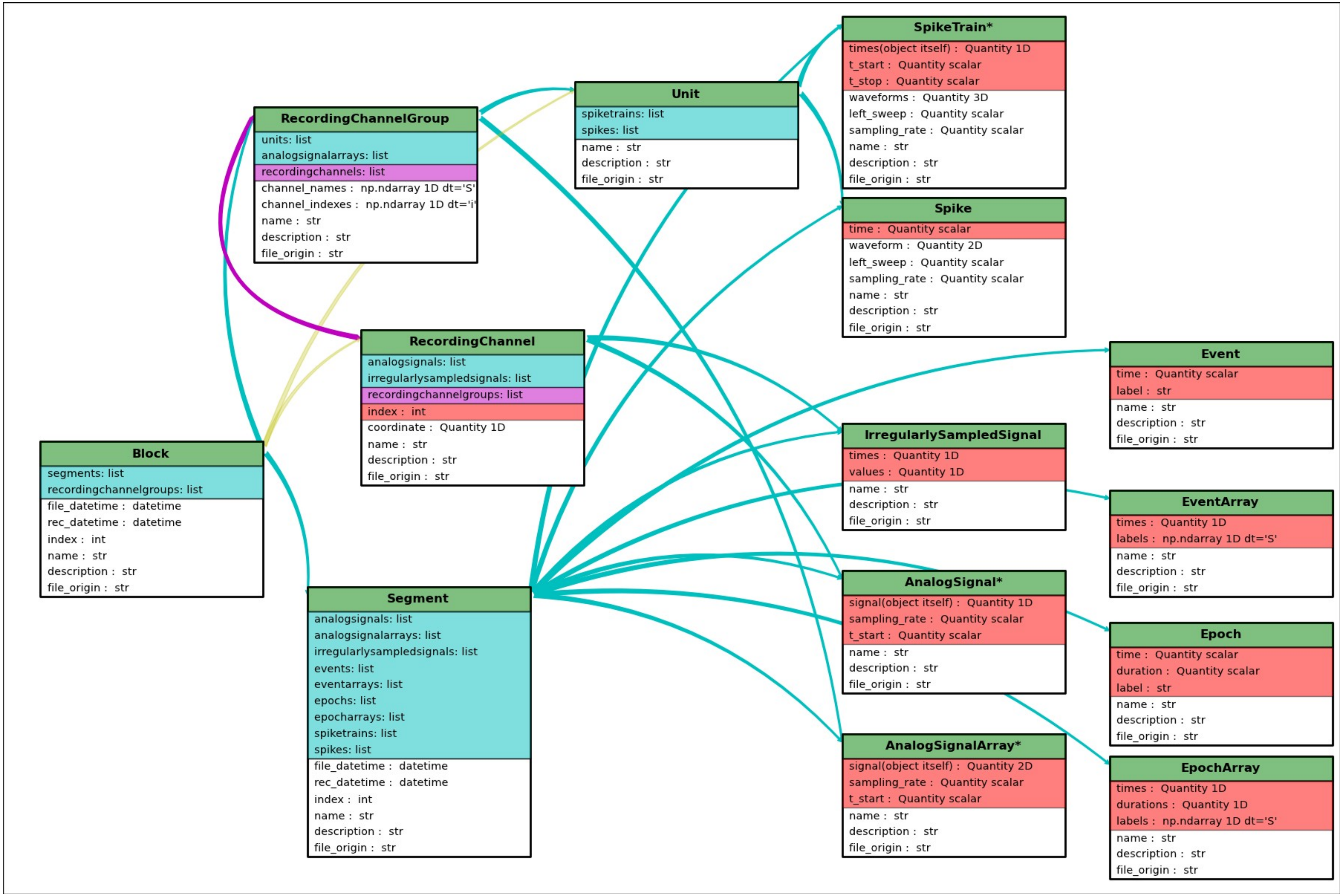
All object have 3 types of attributes:
- Required (AnalogSignal.sampling_rate, AnalogSignal.t_start, ...)
- Recommended (AnalogSignal.name, ...)
- Free in annotations dict:

```
>>> seg = Segment()
>>> seg.annotate(stimulus="step pulse", amplitude=10*nA)
>>> print(seg.annotations)
{'amplitude': array(10.0) * nA, 'stimulus': 'step pulse'}
```

SpikeTrain, AnalogSignal, and AnalogSIgnalArray inherits python-quantities:directly behave like np.array with units.

```
>>> import neo
>>> st = neo.SpikeTrain([3, 4, 5], units='sec', t_stop=10.0)
>>> print(st)
[ 3.  4.  5.] s
```

Class tour: schema



Class tour : definition

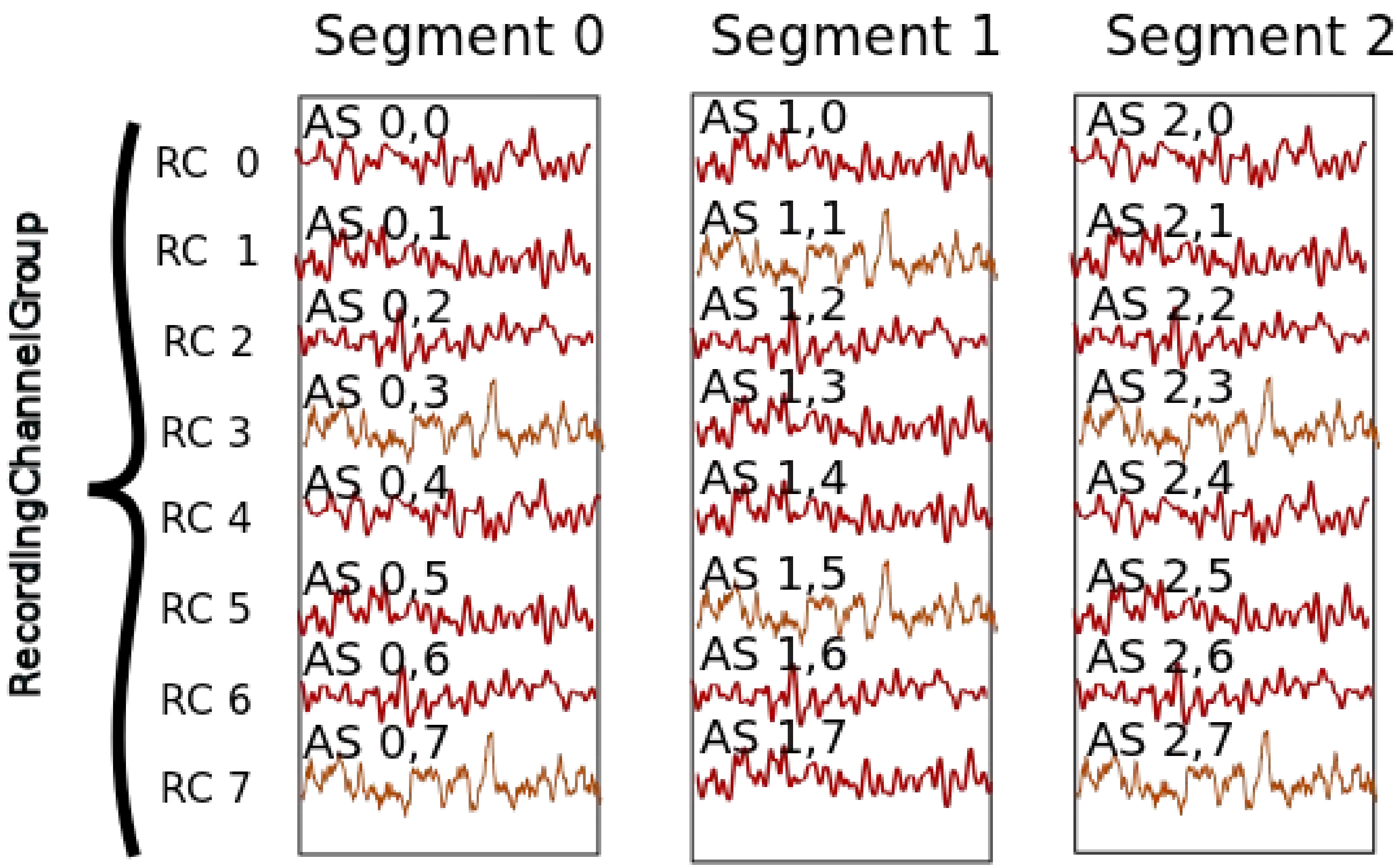
AnalogSignal: A regular sampling of a continuous, analog signal.
AnalogSignalArray: A regular sampling of a multichannel continuous analog signal. (2D NumPy array)
Spike: One action potential characterized by its time and waveform.
SpikeTrain: A set of action potentials (spikes) emitted by the same unit in a period of time (with optional waveforms).
Event and EventArray: A time point representng an event in the data, or an array of such time points.
Epoch and EpochArray: An interval of time representing a period of time in the data, or an array of such intervals.

Segment: A container for heterogeneous discrete or continous data sharing a common clock (time basis)
but not necessarily the same sampling rate, start time or end time. A Segment can be considered as equivalent
to a "trial", "episode", "run", "recording", etc., depending on the experimental context. May contain any of the data objects.
Block: The top-level container gathering all of the data, discrete and continuous, for a given recording session.
Contains Segment and RecordingChannelGroup objects.

RecordingChannelGroup: A group for associated RecordingChannel objects. This has several possible uses:
RecordingChannel objects of the same array.
Unit: A Unit gathers all the SpikeTrain objects within a common Block, possibly across several Segments,
that have been emitted by the same cell. A Unit is linked to RecordingChannelGroup objects from which it was detected.
This replaces the Neuron class in the previous version of Neo (v0.1).

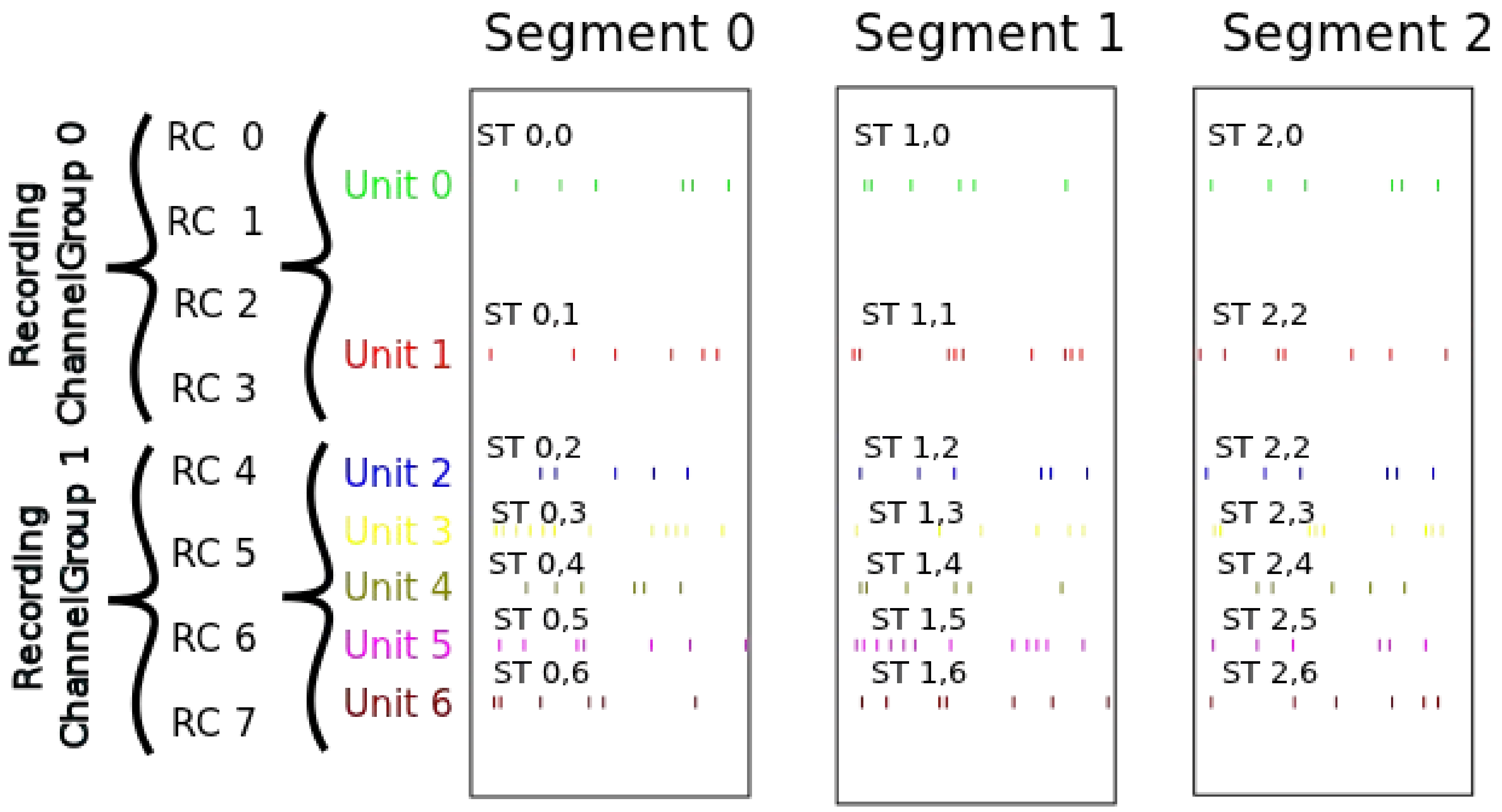
Class tour : Use case

RC = RecordingChannel
AS = AnalogSignal



Class tour : Use case

RC = RecordingChannel
ST = SpikeTrain



IO tour

First interest to have same classes :
Same API to read/write data files.

All formats are really different so we need a flexible API:
• ABF = Block+Segment+AnalogSignal+Event
• Plexon = Segment+SpikeTrain+Spike+AnalogSignal
• PyNN = SpikeTrain+AnalogSignal
• RAW = AnalogSignal

What is this API ?
• For each format you have an IO class
• The IO class can read or write one or several neo objects.

IO : tour

Module	Python 2	Python 3
AlphaOmegaIO	Yes	No
AsciiSignalIO	Yes	Yes
AsciiSpikeTrainIO	Yes	Yes
AxonIO	Yes	No
BlackrockIO	Yes	No
ElanIO	Yes	No
HDF5IO	Yes	No
KlustakwikIO	Yes	No
MicromedIO	Yes	No
NeoMatlabIO	Yes	Yes
NeuroExplorerIO	Yes	No
PlexonIO	Yes	No
PyNNIO	Yes	Yes
RawBinarySignalIO	Yes	Yes
Spike2IO	Yes	Yes
TdtIO	Yes	No
WinEdrIO	Yes	Yes
WinWcpIO	Yes	Yes

IO tour : workflow

One class per format:

```
>>> from neo.io import MyFormatIO
>>> reader = MyFormatIO(filename = "myfile.dat")
```

Different modes (file, dir, database, ...)

```
>>> from neo.io import MyFormatIO
>>> print MyFormatIO.mode
'file'
```

Examples

```
>>> reader = io.PlexonIO(filename='File_plexon_1.plx')
>>> reader = io.TdtIO(dirname='aep_05')
```

IO tour : workflow

Concept of readable/supported objects:

```
>>> MyFormatIO.supported_objects
[Segment , AnalogSignal , SpikeTrain, Event, Spike]
```

```
>>> MyFormatIO.readable_objects
[Segment]
```

Class offer reading method for readable objects

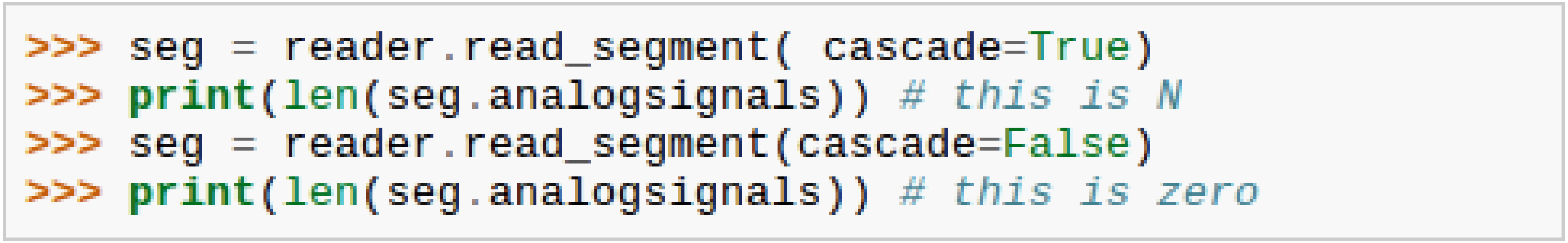
```
>>> seg = reader.read_segment()
>>> type(seg)
neo.core.Segment
```

All classes propose read() = read_block()

```
>>> bl = reader.read()
>>> print bl.segments[0]
neo.core.Segment
```

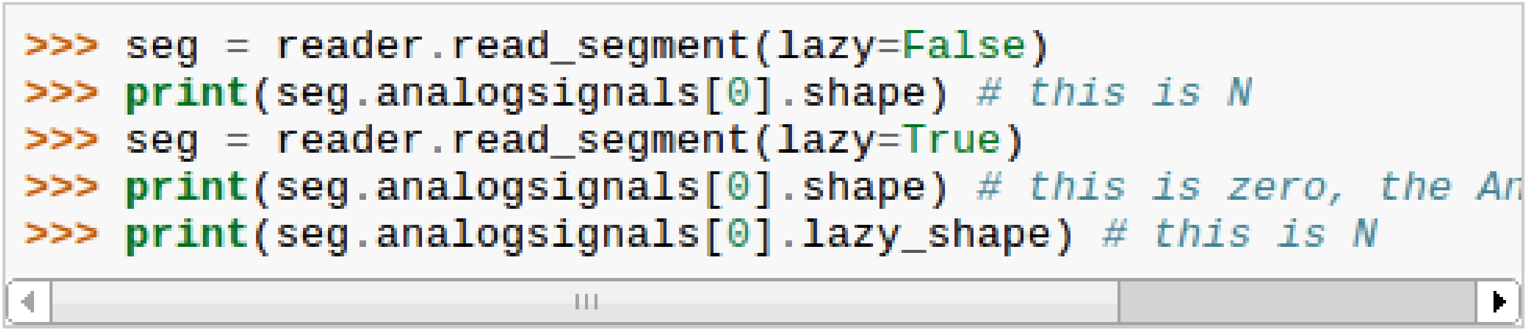
IO tour : workflow

Cascade option:



```
>>> seg = reader.read_segment( cascade=True)
>>> print(len(seg.analogsignals)) # this is N
>>> seg = reader.read_segment(cascade=False)
>>> print(len(seg.analogsignals)) # this is zero
```

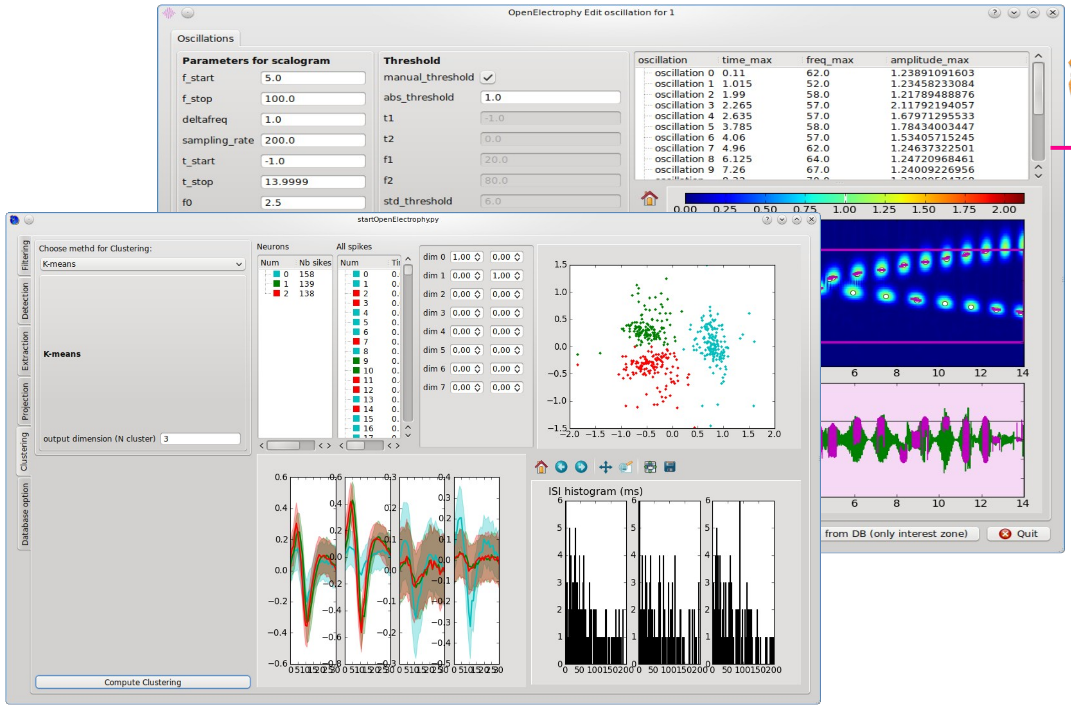
Lazy option:



```
>>> seg = reader.read_segment(lazy=False)
>>> print(seg.analogsignals[0].shape) # this is N
>>> seg = reader.read_segment(lazy=True)
>>> print(seg.analogsignals[0].shape) # this is zero, the An
>>> print(seg.analogsignals[0].lazy_shape) # this is N
```

Projects on top of neo

OpenElectrophy



OpenElectrophy Edit oscillation for 1

Oscillations

Parameters for scalogram

f_start	5.0
f_stop	100.0
deltafreq	1.0
sampling_rate	200.0
t_start	-1.0
t_stop	13.9999
f0	2.5

Threshold

manual_threshold	✓
abs_threshold	1.0
t1	-1.0
t2	0.0
f1	20.0
f2	80.0
std_threshold	6.0

oscillation	time_max	freq_max	amplitude_max
oscillation 0	0.11	62.0	1.23891091603
oscillation 1	1.015	52.0	1.23458233084
oscillation 2	1.99	58.0	1.21789488876
oscillation 3	2.265	57.0	2.11792194057
oscillation 4	2.635	57.0	1.67971295533
oscillation 5	3.785	58.0	1.78434003447
oscillation 6	4.06	57.0	1.53405715245
oscillation 7	4.96	62.0	1.24637322501
oscillation 8	6.125	64.0	1.24720968461
oscillation 9	7.26	67.0	1.24009226956

startOpenElectrophy.py

Choose methd for Clustering: K-means

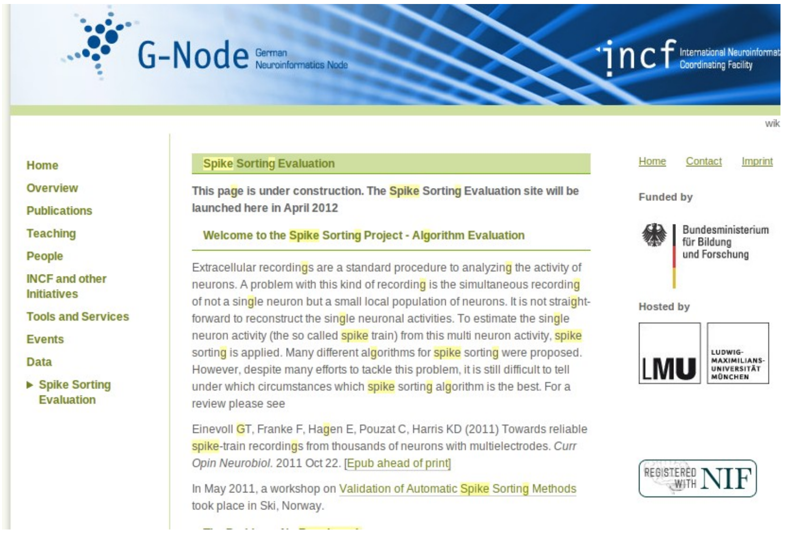
K-means

output dimension (N cluster) 3

Neurons

Num	Nb sikes
0	158
1	139
2	138

Web portal for benchmarking spike sorting algorithm
At G-Node (Felix Franke, Andrey Slobodev)



G-Node German Neuroinformatics Node

incf International Neuroinformat Coordinating Facility

Home
Overview
Publications
Teaching
People
INCF and other Initiatives
Tools and Services
Events
Data
▸ Spike Sorting Evaluation

Home Contact Imprint

Spike Sorting Evaluation

This page is under construction. The Spike Sorting Evaluation site will be launched here in April 2012

Welcome to the Spike Sorting Project - Algorithm Evaluation

Extracellular recordings are a standard procedure to analyzing the activity of neurons. A problem with this kind of recording is the simultaneous recording of not a single neuron but a small local population of neurons. It is not straight-forward to reconstruct the single neuronal activities. To estimate the single neuron activity (the so called spike train) from this multi neuron activity, spike sorting is applied. Many different algorithms for spike sorting were proposed. However, despite many efforts to tackle this problem, it is still difficult to tell under which circumstances which spike sorting algorithm is the best. For a review please see

Einevoll GT, Franke F, Hagen E, Pouzat C, Harris KD (2011) Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Curr Opin Neurobiol.* 2011 Oct 22. [Epub ahead of print]

In May 2011, a workshop on Validation of Automatic Spike Sorting Methods took place in Ski, Norway.

Funded by
Bundesministerium für Bildung und Forschung

Hosted by
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

REGISTERED WITH NIF

New NeuroTools

Mozaik

PyNN

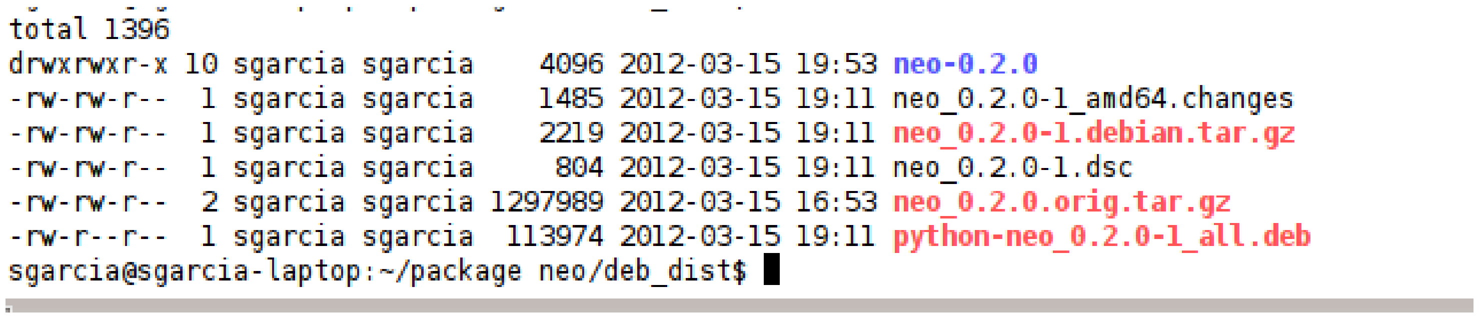
Conclusion

- If your project generate data : write an IO for neo
- If your project manage signals and spikes : provide an interface to neo objects
- If your experimentalist colleague wants to read data set from commercial systems: neo.io

Thanks to the the neo team

Samuel Garcia
Andrew Davison
Chris Rodgers
Pierre Yger
Yann Mahnoun
Luc Estabanez
Andrey Sobolev
Thierry Brizzi
Florent Jaillet
Philipp Rautenberg
Thomas Wachtler
Cyril Dejean

Thanks to Michael Hanke to make the very first debian package of my life.



```
total 1396
drwxrwxr-x 10 sgarcia sgarcia    4096 2012-03-15 19:53 neo-0.2.0
-rw-rw-r--  1 sgarcia sgarcia    1485 2012-03-15 19:11 neo_0.2.0-1_amd64.changes
-rw-rw-r--  1 sgarcia sgarcia    2219 2012-03-15 19:11 neo_0.2.0-1.debian.tar.gz
-rw-rw-r--  1 sgarcia sgarcia     804 2012-03-15 19:11 neo_0.2.0-1.dsc
-rw-rw-r--  2 sgarcia sgarcia 1297989 2012-03-15 16:53 neo_0.2.0.orig.tar.gz
-rw-r--r--  1 sgarcia sgarcia  113974 2012-03-15 19:11 python-neo_0.2.0-1_all.deb
sgarcia@sgarcia-laptop:~/package neo/deb_dist$
```