From the workbench: Tools for interacting with structured neurophysiological data in the G-Node storage system

TBD

August 8, 2011

Structured, efficient and secure storage of experimental data and associated meta-information constitutes one of the most pressing challenges in modern neuroinformatics, and does so particularly in electrophysiology. The German INCF node aims to provide a full-stack solution for this specific domain, consisting of server-side infrastructure that holds data in an object model tailored to field-relevant demands and exposes these entities through a convenient HTTP-based RESTful interface, as well as a collection of client-side tools which facilitate data consumption and enable integration into various popular analysis environments such as MATLAB, Python and R. Existing workflows and code are leveraged whenever possible: By transparently managing the transition from G-Node object format to native data structures and vice versa, our client libraries allow scientists to benefit from a powerful, semantic storage system without having to drastically alter established processes.

The object model, derived from Python's NEO, offers a carefully designed set of abstractions capturing the most prevalent components of electrophysiological recordings and ranging from high-level containers (e.g., blocks or segments) down to fine-grained elements (e.g., spikes, waveforms or raw signal data). In combination with the light-weight mark-up language odML, this model represents a comprehensive approach to describing electrophysiological data and attached meta-data. In order to maximize client compatibility, access to these objects is primarily mediated by a concise, RESTful API which transmits data via HTTP and in form of machine- as well as human-readable JSON objects. Any programming and analysis suite capable of performing simple HTTP requests therefore gains the ability to consume, modify and create data maintained in the G-Node storage system.

Given the variety of analysis environments in use, we acknowledge that a sensible storage solution needs to integrate with existing toolchains, and must do so seamlessly. To this end, G-Node puts forward a set of client libraries for Python, JVM languages (e.g., Java), MATLAB and R which make stored recordings readily accessible from each ecosystem and hide virtually all required middleware operations (e.g., transfer, parsing, caching, and so on). Key aim is a natural user experience, adhering to each language's idiosyncrasies and transforming G-Node objects into idiomatic and computationally efficient local representations. For instance, while the Python interface implements a close mapping between stored and local objects, our MATLAB toolbox exploits the runtime's heavily optimised array handling when rendering objects. Bindings to R, on the other hand, present requested data as suitably laid out data frames. By employing standard patterns in each environment, researchers retain straightforward access to relevant libraries supplied by their respective analysis tools. Transformations of this kind are bidirectional: client libraries facilitate tagging and storing of existing recordings and support re-upload of modified objects. A powerful side-effect of this unified data store is therefore enhanced inter-operability between languages—recordings are easily manipulated by disparate client applications. In addition, we are currently developing browser-based visualisation tools aiding scientists in exploratory data analysis.

Crucially, the extent to which advanced features of the object model are used is user-determined; each library's design emphasises the importance of gradual integration. Thus, by providing unobtrusive yet fully-featured access, our client utilities bring the advantages of the G-Node data management system closer to the electrophysiologist's workbench.