PyGêBR: a Python library for dealing with processing flows

R. Biloti, University of Campinas & INCT-GP/CNPq, Brazil and E. F. F. Silva, Petrobras, Brazil

This paper was prepared for presentation during the 18th International Congress of the Brazilian Geophysical Society held in Rio de Janeiro, Brazil, 16-19 October 2023.

Abstract

PyGêBR is a free-software library to represent chains of command-line programs through which data flows. That's commonly refered as *processing flows*. Processing flows are usually employed to process scientific data, like seismic data, which was the original field for which PyGêBR was written for.
Beyond representing processing flows, by means of PyGêBR, one can inspect, build, edit, merge, display, share and run processing flows from within Python scripts.

PyGêBR is an evolution of original GêBR interface, that dates back to 2005~2007. At that time, Petrobras fostered the development of GêBR through a project supported by its research net on Geophysics. That interface was written in C with GTK2 and played an important role in teaching seismic processing in undergraduate Geophysical courses in Brazil. Unfortunately, it's maintenance proved to be cumbersome. Without an active community of developers, its legacy code is hardly built today.

In 2021, PyGêBR started to be written from scratch as a Python library, having a diverse publice as target audience, sharing Python as baseline knowledge. That choice reoriented the project focus. A graphical interface to operate with processing flows is no longer the main point. Instead, PyGêBR library provides the means to programmatically represent and deal with processing flows in a Python native way. In this sense, one can build processing flows with command-line programs mixed with Python code.

A Python schema is defined for each flow component, from single command-line arguments, or *parameters*, to *programs* and chains of programs, or *flows*. Besides, the library provides methods to deal with each of these entities. These back-end functionalities are combined to build high level functions to assemble processing flows from predefined blocks, or *menus*, representing command-line programs, edit and inspect them, run and even access their outcomes from within Python code.

As *Python notebooks* are common ways of organize, document, run and share code blocks, it worths to use PyGêBR within such environment. In this regard, a set of functions is also provided to be used in notebooks. In special, a simple but practical graphical interface can be built in real time for each flow. That permits easy and fast setup of processing flow's parameters and allows to communicate and/or export such setup in a clear way to others.

To conclude, PyGêBR code is much simpler than its ancestor GêBR. It is more accessible to contributors and easier to maintain. We envision that this characteristics will allow the grow of community of users/developers to sustain the PyGêBR project longer.