PSYC 5720 — Computational Neuroscience

Data Exploration Assignment: Reading the Data

April 26, 2022

The goal of this assignment is to start work on the infrastructure for reading your data into Python. There are really two parts to this problem. One is about how to access the data as it's stored on the disk; the other is about how to represent the data in memory so that you can easily inspect and manipulate it.

At this point you need to have settled on which CRCNS data set you'll be analyzing.

Setting up your workspace

You'll be storing and submitting your work on this project through on Rivanna.

1. Create a subdirectory on Rivanna under `/projects/psyc5270-cdm8j/data-exploration`. Give it a name consisting of the computing ids of all the members of your group, separated by dashes (e.g. `cdm8j-ckf5h`)

Document data retrieval

Move your CRCNS data set into the project directory under the directory `data`.

Do not check your data into git. Instead, edit `data/README.md` to describe which files you retrieved from the dataset and any additional steps you took to organize them, including files or directories that you are not using.

Initialize your computing environment

Follow the instructions in the top-level `README.md` on how to set up your computing environment. It's critical to document your software dependencies and keep them isolated from system packages.

Write a module to load your data

The main part of this assignment is to write one or more functions to load your data into Python. You can write this code directly in `src/io.py`. If you prefer to work in a notebook, start the notebook server with `jupyter-server` and create a notebook at the top level. You can later copy your function to `src/io.py`.

This requires some planning, which you should have already done to some extent at the end of Lesson 6B. You need the following:

- a general idea of your hypothesis and the specific question you want to answer
- a determination of what the natural unit of analysis (UoA) is for your study
- knowledge of how the relevant data associated with your UoA is organized in the files of the data repository

You now need to write a function that will pull the data from the files into Python and organize it so that it's easy to work with. This function will look something like this:

```
def read_neuron(id, param_1=default_1):
    """Describe what the arguments of the function mean and what the function returns"""
```

```
body_of_the_function
```

You can copy this *stub* to `src/io.py` to get started or to your notebook. Before you start writing code, think about what the arguments to the function should be and what it should return (i.e., write the docstring first). This is called the function's *interface*.

Decide how your UoAs will be identified (the `id` argument). The data set may already have names for them, or you may need to assign numerical labels. Identify any additional arguments that you might need to specify how the data will be processed. It's good to keep these to a minimum, however.

You might need to hard-code certain variables, like the names of directories where certain kinds of data are located. This is fine, because your code is only designed to work with this data set.

Don't reinvent the wheel. Make sure to check if one of the following packages already has a function to do what you want:

- Standard Python library:
  - os: construct path names
  - glob: find files using wildcards
  - json: read data encoded in JSON format
- Numpy:
  - loadtxt: read numerical data in text format into numpy arrays
  - memmap: read numerica data in binary format into numpy arrays
- Scipy:
  - io: read and write data in a variety of formats
- Pandas:
  - io: read and write structured data formats (including CSV and Excel files)
- 3rd party packages
  - neo.io: Many widely used electrophysiology formats
  - nibabel: Many widely used imaging formats
  - use your favorite search engine to find others

Keep in mind that this is a work in progress. The most important part of the task is to write a simple, clear interface, and then try to implement it as simply as possible. You will certainly have to add more functionality later, so keeping things simple and straightforward will pay off big dividends down the line.

You will have an opportunity to work on this for one class period next week.

Push your work to github

Stage any files you created or changed with `git add`. You can use `git status` to determine what files have been added or changed. For example:

```
git add mynotebook.ipynb README.md data/README.md setup.py requirements.txt
```

Commit the staged changes with `git commit`. Write a short message describing what changed:

```
git commit -m "renamed project and added read_neuron to io"
```

Push the changes to github:

```
git push
```

Evaluation

Your score for this assignment will be based on whether you followed the instructions in this document and in the `README.md` file of the `comp-neurosci-skeleton` repository. Particular attention will be paid to the following:

- data files are not checked in to the repository
- instructions for retrieving data are in the `data/README.md` file
- `io.py` module is present and contains a working function to load part or all of the data