Array programming for Biology

This manuscript (permalink) was automatically generated from bionumpy/bionumpy-correspondence@9a596e5 on December 6, 2023.

Authors

- **Knut Dagestad Rand** ✉
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway; Centre for Bioinformatics, University of Oslo, Oslo, Norway

- **Ivar Grytten**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway

- **Milena Pavlović**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway

- **Chakri**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway

- **Geir Kjetil Sandve**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway; Centre for Bioinformatics, University of Oslo, Oslo, Norway; UiORealArt Convergence Environment, University of Oslo, Oslo, Norway

✉ — Correspondence possible via GitHub Issues or email to Knut Dagestad Rand <knutdr@math.uio.no>.

Array Programming for Biology

Knut Rand [1]_, Ivar Grytten, Milena Pavlovic, Chakravarthi Kanduri and Geir Kjetil Sandve

.. [1] Correspondence: knutdr@ifi.uio.no

To the editor: Python is a widely used programming language for scientific computing, in large part due to the powerful *array programming* library NumPy [**numpy?**], which makes it easy to write clean, vectorized and computationally efficient code for handling large datasets. A challenge with using array programming in biology is that the data is often non-numeric and variable-length (e.g. DNA sequences), inhibiting out-of-the-box use of standard array programming techniques.This may push bioinformaticians to instead rely on complex, custom pipelines of UNIX commands that are non-transparent and error-prone. Furthermore, the impracticality of developing efficient code directly in high-level languages like Python has led to tool developers almost exclusively relying on low-level languages like C and C++, making it more difficult for computational biologists to understand and contribute to core methods in the field.

We present the BioNumPy package, which enables efficient and intuitive array programming on biological data in Python. This is achieved by storing biological datasets in data structures built on top of underlying NumPy arrays (see Supplementary Material). BioNumPy supports a broad range of bioinformatics analyses, with the main philosophy being that data structures should behave as closely as possible to standard numeric NumPy arrays. This means that BioNumPy is easy to learn for users familiar with NumPy or with array programming languages like R and Matlab. BioNumPy is open-source and freely available at https://github.com/bionumpy/bionumpy/, and can be installed through the Python package manager Pip. BioNumPy comes with extensive documentation and a user guide that makes it easy to use for a wide range of molecular biology datasets and problems.

BioNumPy is able to read and write biological datasets (e.g FASTQ, BAM, or VCF-files) directly to/from NumPy-like data structures, providing efficient access to the data through an intuitive and easy-to-use API. The data is then processed and analysed efficiently using a NumPy-like interface.

In the following example, we showcase BioNumPy by reading sequenced reads from a FASTQ file and plotting the average base quality per read position. Both the sequences and base qualities are represented in NumPy-compatible arrays, so that NumPy-functionality like e.g. np.mean can be used.

.. code-block:: python

```
reads = bionumpy.open("reads.fq.gz").read_chunk() mean_qual_per_base =
numpy.mean(reads.quality, axis=0) plotly.express.line(mean_qual_per_base[0:150]).show()
```

Figure 1: An example of using BioNumPy with the resulting plot from running the code.

We show through a range of experiments that BioNumPy is considerably faster than existing Python packages for common bioinformatics tasks and, in many cases, as fast as tools written in C/C++ (see Supplementary Material). We also showcase how BioNumPy enables seamless machine learning on biological sequence data by reproducing parts of a recent machine learning benchmark study using only X lines of code (as compared to Y lines in the original implementation). (Supplementary Material).

In conclusion, we believe that BioNumPy bridges a long-lasting gap by making array programming practical for the field of biology.

References