dinopy: Efficient DNA Input and Output with Cython

Henning Timm and Till Hartmann

Genome Informatics; Institute of Human Genetics; University Hospital Essen; University of Duisburg-Essen, Germany

Introduction - Why create another IO library?

Dinopy (Dna INput and Output in PYthon) is a Python package that aims to simplify the development of bioinformatics applications by providing efficient facilities for DNA input and output. At the time of writing, there is no library for I/O of DNA specific files available which makes full use of the potential of Cython [1].
Dinopy exports Cython level API bindings which can be used by other Cython applications for increased speedup.

The Power of Cython

Cython aims to combine the performance of C and C++ with Python's convenient development process. This is achieved by translating Cython code to C and compiling it.

Advantages:

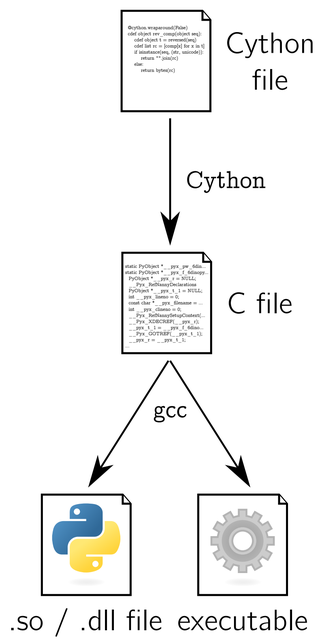
- fewer type checks at runtime
  - fast numeric operations
  - faster execution
- easy integration of efficient C / C++ code
- used in many big projects (SciPy [3], pandas [4], scikit-learn [6], ...)

Downsides:

- manual typing required
- not all Python features supported (yet)

Goals:

- early typing → avoid Python overhead
- use efficient C level functions wherever possible



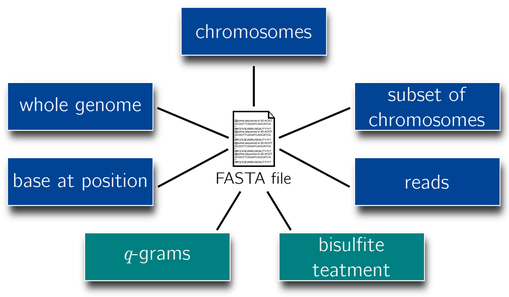
Core Goals of dinopy

Goal 1 - performance:

- data type (dtype) system, similar to numpy [8]
- efficient (C level) functions to modify sequences (tailored to dtypes)
- generators → small memory footprint
- 2bit and 4bit encoding of sequences
- use faidx index files → random access on FASTA files
- provide Cython level API bindings

Goal 2 - usability:

- different views on sequences
- modular processors to modify views and sequences
- exclusion of information (e.g. read names, quality values) when not needed
- automatic handling of gzipped files



Availability

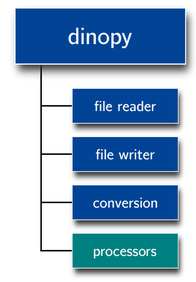
The dinopy package is available for Python 3.2+ under the MIT License. It has so far only been tested on Linux operating systems.

The source code can be downloaded from Bitbucket (`https://bitbucket.org/HenningTimm/dinopy` or scan the QR code on the right) and installed using `python setup.py install`.

Overview

The dinopy package can be structured into:

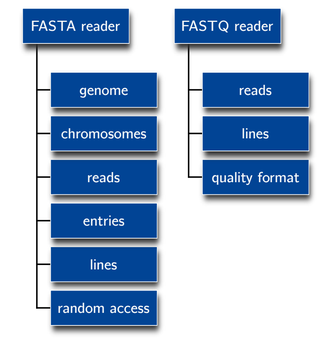
- **reader** for FASTA and FASTQ files
- **writer** for FASTA and FASTQ files
- **conversion** functions for sequence data types and quality value formats
- **processors** for the analysis of sequences



File Reader

Reader classes which provide different views on the opened files.

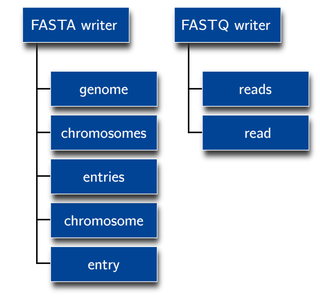
- view can be chosen to best fit analysis
- random access → efficient extraction of sequences from big FASTA files
  - chromosomes
  - single bases and subsequences
- automatic estimation of quality formats



File Writer

Writer classes which allow to write to files as well as to streams.

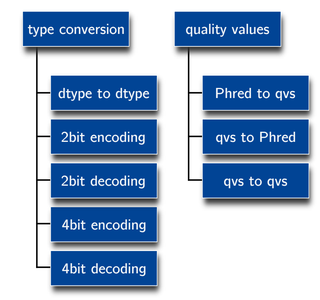
- support bulk writing and iterative writing
- allow stream processing (reading from `stdin` and writing to `stdout`)
- automatic creation of faidx files for written FASTA files



Conversion

Conversion of dtypes and quality formats:

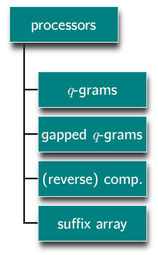
- dtype to dtype conversion
- memory efficient 2bit and 4bit encoding
- Phred quality scores ↔ quality values
- quality value systems (Illumina 1.3, 1.5, 1.8; Solexa; Sanger)



Processors

Functions for sequence processing:

- creation of q-grams from sequences
  - solid
  - gapped / shaped (`##--##-#`, `ACGTACGT` → `ACACT`)
- (reverse) complement for all dtypes and encodings
- suffix array (constructed in linear time [5])



Feature Comparison

Dinopy offers a unique combination of features:

	dinopy	BioPython [2]	fastAQ[1]	pyfaidx [7]	pysam[2]
FASTQ input	✓	✓	✓		✓
faidx support	✓			✓	✓
gzip support	✓	(✓)	(✓)		✓
variable data types	✓			(✓)	
Cython API	✓				(✓)

[1] `https://github.com/Jverma/fastAQ/`

[2] `https://github.com/pysam-developers/pysam`

References

[1] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[2] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[3] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-09-26].

[4] W. McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

[5] G. Nong, S. Zhang, and W. H. Chan. Two efficient algorithms for linear time suffix array construction. *Computers, IEEE Transactions on*, 60(10):1471–1484, 2011.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] M. D. Shirley, Z. Ma, B. S. Pedersen, and S. J. Wheelan. Efficient "pythonic" access to fasta files using pyfaidx. Technical report, PeerJ PrePrints, 2015.

[8] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.