SOFTWARE TOOL ARTICLE

REVISED pyGeno: A Python package for precision medicine and proteogenomics [version 2; referees: 1 approved, 2 approved with reservations]

Tariq Daouda[1,2], Claude Perreault[1,3,4], Sébastien Lemieux[1,5]

[1]Institute for Research in Immunology and Cancer, Université de Montréal, Montreal, Canada
[2]Department of Biochemistry, Faculty of Medicine, Université de Montréal, Montreal, Canada
[3]Division of Hematology, Hôpital Maisonneuve-Rosemont, Montreal, Canada
[4]Department of Medicine, Faculty of Medicine, Université de Montréal, Montreal, Canada
[5]Department of Computer Science and Operations Research, Faculty of Arts and Sciences, Université de Montréal, Montreal, Canada

v2 **First published:** 21 Mar 2016, **5**:381 (doi: 10.12688/f1000research.8251.1)
Latest published: 10 May 2016, **5**:381 (doi: 10.12688/f1000research.8251.2)

Abstract

pyGeno is a Python package mainly intended for precision medicine applications that revolve around genomics and proteomics. It integrates reference sequences and annotations from Ensembl, genomic polymorphisms from the dbSNP database and data from next-gen sequencing into an easy to use, memory-efficient and fast framework, therefore allowing the user to easily explore subject-specific genomes and proteomes. Compared to a standalone program, pyGeno gives the user access to the complete expressivity of Python, a general programming language. Its range of application therefore encompasses both short scripts and large scale genome-wide studies.

Open Peer Review

Referee Status: ? ✓ ?

	Invited Referees 1	2	3
REVISED **version 2** published 10 May 2016	? report	✓ report	? report
version 1 published 21 Mar 2016			

1 **Lynn Fink**, University of Queensland Australia

2 **Hilary Coller**, University of California Los Angeles USA

3 **Christian Cole**, University of Dundee UK

Discuss this article

Comments (0)

Corresponding authors: Tariq Daouda (tariq.daouda@umontreal.ca), Sébastien Lemieux (s.lemieux@umontreal.ca)

How to cite this article: Daouda T, Perreault C and Lemieux S. **pyGeno: A Python package for precision medicine and proteogenomics [version 2; referees: 1 approved, 2 approved with reservations]** *F1000Research* 2016, **5**:381 (doi: 10.12688/f1000research.8251.2)

Grant information: This work was supported by the Canadian Cancer Society (Grant number 701564), assigned to Claude Perreault.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 21 Mar 2016, **5**:381 (doi: 10.12688/f1000research.8251.1)

REVISED **Amendments from Version 1**

Updated the archived source code link to correspond to the last version since the original publication 1.2.8.

See referee reports

Introduction

High-throughput systems biology and precision medicine applications require the integration of data from many different sources. For instance, a significant part of precision medicine research revolves around the identification of relevant single nucleotide polymorphisms (SNPs) and insertions/deletions (INDELS) and the study of their context[1]. Furthermore recent studies in proteogenomics show that replacing traditional reference databases such as Uniprot[2] by customized databases that integrate the subject's genomic polymorphisms, can significantly improve the identification of peptides or proteins using mass spectrometry[3–6]. These applications usually require the integration of reference sequences, reference genome annotations, specific SNPs and INDELs along with an external SNP database such as dbSNP[7] for validation. The sheer amount of data generated by theses studies rules out most spreadsheet analyses and requires tools that are both fast and memory efficient. Furthermore, these studies often require the collaboration of people with different sets of skills. Thus, it was important to us to develop a tool that is powerful enough to be integrated in complex high-throughput pipelines, while still being understandable by users with limited technical abilities. In contrast to other projects such as BioPython[8] and PyCogent[9] whose objective is to provide a general set of tools for bioinformatics, the primarily ambition behind pyGeno is to provide the community with a powerful genome and proteome exploration tool that can be easily integrated into scripts. The current version integrates gene set annotations and reference sequences from Ensembl[10] along with polymorphisms (both SNPs and INDELs) derived from dbSNP[7], and experimentally detected patient-specific polymorphisms.

To our knowledge pyGeno is the only available tool that provides this kind of integration in an easy-to-use and programming-friendly environment. Furthermore, more advanced users can rely on object-oriented inheritance to extend the functionalities of pyGeno to implement support for polymorphisms from other sources. pyGeno has been used with human and mouse genomes and should readily work with any diploid organism whose annotations are made available by Ensembl.

Methods

Design and implementation

pyGeno is written in Python, a language that enjoys a large set of well established and mature scientific libraries that are used in research fields such as physics, mathematics and bioinformatics[8,11–13]. pyGeno gives users access to the full expressivity of Python to explore reference and patient-specific genomes and proteomes, by manipulating familiar objects such as genomes, chromosomes, genes, transcripts, proteins and exons. In order to make pyGeno as easy to use and learn as possible, we have created an interface where only one function, *get()*, can be used for almost any query. An example of usage can be seen in Figure 1. An integrated documentation is also available through the *help()* function.

The current version of pyGeno does not require any access to remote REST APIs. This results in more robust and faster processing since the application is not affected by connection speed or sudden changes to the server API. On the other hand it also implies that extra care must be taken regarding the optimization of the application.

Memory efficiency and speed are mainly achieved through the use of a custom lazy object-oriented database system that we have specifically written for pyGeno (https://github.com/tariqdaouda/rabaDB). When an object is loaded through the *get()* function, only a minimal version of it is served. The object fully develops only once the user accesses a field that is not present in the minimal version (Figure 1). The transformation is entirely transparent and does not require more memory than necessary to store the fully developed object. This is especially important, since most of the time users are only interested in specific regions of the genome, and do not require that the full genome be loaded into memory. Every loaded object is also a singleton, if the user asks for a previously loaded object, pyGeno will serve the object in memory.

Furthermore, this database system is built on top of SQLite version 3 (http://www.sqlite.org/), a serverless relational database. Because SQLite3 uses single files to store data, pyGeno's database can be easily backed up and shared by a simple copy/paste. Moreover, the files can be directly read, modified and analyzed through any SQLite3 client.

As with any other database system, indexes play a crucial role in determining the general performance. Within pyGeno's database, several reference genomes along with patient-specific data and versions of dbSNP can coexist. Therefore building indexes for all the stored information would result in unnecessarily large databases. We therefore have taken the approach of giving the end user full control over indexation through the *ensureGlobalIndex()* and *dropGlobalIndex()* functions. Users can, for example, decide to index the field *'id'* of transcripts by using *Transcript.ensureGlobalIndex('id')* and dramatically improve queries based on transcript ids.

pyGeno's database is populated through imports of datawraps using *importSNPs* and *importGenome* functions. Datawraps are compressed archives that can be shared among co-workers, and are designed to solve the version and update problems. A datawrap contains at least one file named manifest.ini that contains basic information about the package such as a description, a version and a maintainer, as well a list of files from which data must be extracted. It is possible to either compress these files within the archive, or to specify URLs from which the files can be downloaded.

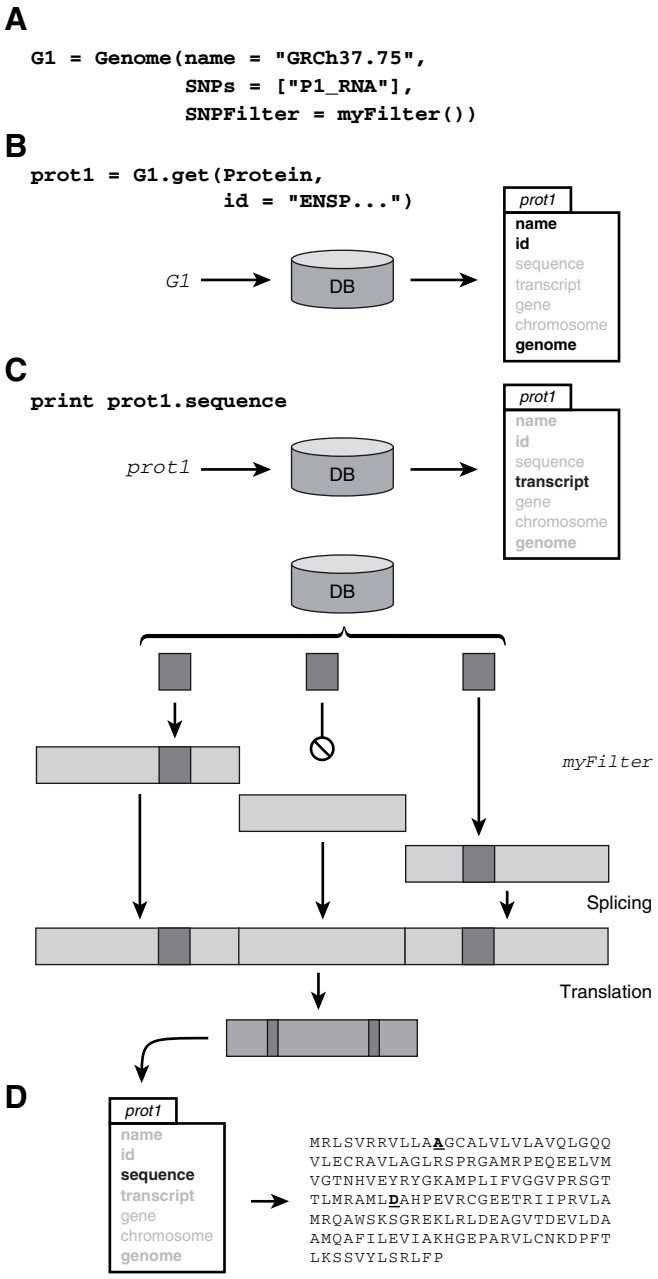

Figure 1. Extracting the subject-specific sequence of a protein. (**A**) Here we instantiate a personalized genome G1 by providing the Genome constructor with the name of a reference genome, a set of polymorphisms and a user defined SNP filter (for example a quality filter). (**B**) We then ask the get function of G1 to return a protein by id. The result is an object where only the fields in bold are fully loaded, other fields will be automatically loaded when and if accessed. (**C**) Asking for the currently unloaded sequence of the protein triggers the following sequence of events. The transcript, as well as the exons that encode for it, and any polymorphisms in their regions are loaded. The polymorphisms are filtered according to the filter provided to the genome constructor (for example, according to sequencing quality) and inserted at their corresponding locations. The exons are then assembled into the transcript sequence and the sequence is translated. (**D**) The sequence as well as the transcript are now fully loaded and the sequence of the precision protein is printed.

In an effort to make pyGeno as easy to install as possible we have made it as dependency-free as possible. This approach has motivated our choice for SQLite3, since it is natively supported by Python 2.5 and above, and it also lead us to develop many tools that were subsequently integrated into pyGeno. Among theses tools are various functions for translating sequences, parsers for GTF/GFF, VCF, FASTA, FASTQ and CSV files, a progress bar, and an efficient way of annotating the genome called segment trees.

Personalized genomes

One of the biggest strengths of pyGeno is to allow the user to define personalized genomes. These genomes are built by combining a reference genome with sets of polymorphisms and a filtering function that returns the alleles to be inserted at the appropriate locus (Figure 1). Personalized genomes are a powerful tool that can go beyond the definition of patient-specific genomes. For instance, we recently used this tool to combine the results of both RNA- and DNA-seq data and create more robust personalized genomes that were used to identify protein-derived peptides by mass spectrometry[3]. Furthermore because pyGeno loads the necessary parts of a given reference genome only once, a pyGeno application can handle several personalized genomes without significantly increasing its memory consumption.

Operation

pyGeno's only requirement is Python2 and we highly recommend version 2.7.6 or later. pyGeno can be easily installed using the *pip* package manager (https://pip.pypa.io/) by typing *pip install pyGeno* into command line interface. Alternatively the latest developments can be obtained from the github repository. Once pyGeno's installation has been completed, the first action that users must perform is the importation of a reference genome datawrap. In order to simplify the process pyGeno comes with several datawraps that can be directly listed and installed using its *bootstrap* module. If the desired reference genome is not among the ones provided, users also have the possibility to create their own from scratch by following the steps described in the documentation. After the first reference genome importation, pyGeno is fully functional and users can further expand its database by importing other reference genomes or SNP sets.

Summary

We have developed pyGeno because, in an age where both precision medicine and DNA/RNA sequencing are becoming more and more important, we needed a tool that would allow us to easily work on personalized genomes that include subject-specific genomic features. Nowadays research teams are increasingly multidisciplinary and are composed of people with very different backgrounds. Since we wanted pyGeno to serve as a common language between users, we therefore took great care in making pyGeno easy to install, easy to use and optimized it so it can run on computers with limited resources (eg. laptops). The fact that pyGeno has been downloaded more than 12,000 times over its first year of existence suggests that there is indeed a need for powerful user-friendly precision medicine tools. With pyGeno we have taken a rather unusual approach to user-friendliness. Instead of writing a program with a graphical user interface (GUI), we have decided to create a Python module that fully integrates within the Python environment. This ensures that users can leverage the full expressiveness of Python as well as the functionalities of other python modules such as SciPy and numpy[11],

pandas (http://pandas.pydata.org/) and matplotlib[13], to meet their specific needs. Furthermore, it led us to think of the functions and objects the user manipulates as pyGeno's interface and we strived to make it as simple and easy to learn as possible.

In the past few years great technologies have been developed. Scripting languages such as Python and JavaScript have taken programming to a whole new level of simplicity, and are now fast enough to serve as foundations to large-scale projects. Freely available libraries such as D3.js (http://d3js.org/) allow for the creation of stunning data representations, that once coupled with tools such as pyGeno, could be used to create powerful interactive representations of biological data. The NoSQL movement has produced several new database systems from which developers can choose, offering them the opportunity to store sheer amounts of data with a flexibility that was not present only a few years ago. These technologies and many others are only waiting to be put together into ground breaking tools for the treatment of biological data. In life saving research areas, we believe that great tools that dramatically improve workflow efficiency are not a luxury but a necessity.

Software availability

1. pyGeno is available from the Python Package Index (PyPI; https://pypi.python.org) via: pip install pyGeno.
2. Latest source code: https://github.com/tariqdaouda/pyGeno.
3. Documentation: http://pyGeno.iric.ca
4. Link to archived source code as at time of publication: https://zenodo.org/record/50587#.VyIP0UErJB0 (doi:10.5281/zenodo.50587)
5. License: Apache License Version 2.0

Author contributions

TD designed and developed pyGeno. SL, CP, and TD contributed to the preparation of the manuscript. All authors were involved in the revision of the draft manuscript and have agreed to the final content.

Competing interests

No competing interests were disclosed.

Grant information

This work was supported by the Canadian Cancer Society (Grant number 701564), assigned to Claude Perreault.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

We would like to thank Jean-Philippe Laverdure, Céline Laumont and Hillary Pearson for being the first users (outside of the developer) and the first testers of pyGeno.

References

1. Collins FS, Varmus H: **A new initiative on precision medicine.** *N Engl J Med.* 2015; **372**(9): 793–795.
PubMed Abstract | Publisher Full Text
2. Uniprot Consortium: **Update on activities at the Universal Protein Resource (UniProt) in 2013.** *Nucleic Acids Res.* 2013; **41**(Database issue): D43–47.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Granados DP, Sriranganadane D, Daouda T, *et al.*: **Impact of genomic polymorphisms on the repertoire of human MHC class I-associated peptides.** *Nat Commun.* 2014; **5**: 3600.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Kim MS, Pinto SM, Getnet D, *et al.*: **A draft map of the human proteome.** *Nature.* 2014; **509**(7502): 575–581.
PubMed Abstract | Publisher Full Text | Free Full Text
5. Wilhelm M, Schlegl J, Hahne H, *et al.*: **Mass-spectrometry-based draft of the human proteome.** *Nature.* 2014; **509**(7502): 582–587.
PubMed Abstract | Publisher Full Text
6. Laumont CM, Daouda T, Laverdure JP, *et al.*: **Global proteogenomic analysis of human MHC class I-associated peptides derived from non-canonical reading frames.** *Nat Commun.* 2016; **7**: 10238.
PubMed Abstract | Publisher Full Text
7. Sherry ST, Ward MH, Kholodov M, *et al.*: **dbSNP: the NCBI database of genetic variation.** *Nucleic Acids Res.* 2001; **29**(1): 308–311.
PubMed Abstract | Publisher Full Text | Free Full Text
8. Cock PJ, Antao T, Chang JT, *et al.*: **Biopython: freely available Python tools for computational molecular biology and bioinformatics.** *Bioinformatics.* 2009; **25**(11): 1422–1423.
PubMed Abstract | Publisher Full Text | Free Full Text
9. Knight R, Maxwell P, Birmingham A, *et al.*: **PyCogent: a toolkit for making sense from sequence.** *Genome Biol.* 2007; **8**(8): R171.
PubMed Abstract | Publisher Full Text | Free Full Text
10. Flicek P, Amode MR, Barrell D, *et al.*: **Ensembl 2014.** *Nucleic Acids Res.* 2014; **42**(Database issue): D749–D755.
PubMed Abstract | Publisher Full Text | Free Full Text
11. Jones E, Oliphant T, Peterson P, *et al.*: **SciPy: Open source scientific tools for Python.** 2001; [Online; accessed 2016-02-22].
Reference Source
12. SymPy Development Team: **SymPy: Python library for symbolic mathematics.** 2014.
Reference Source
13. Hunter JD: **Matplotlib: A 2d graphics environment.** *Comput Sci Eng.* 2007; **9**(3): 90–95.
Publisher Full Text

Open Peer Review

Current Referee Status: ? ✓ ?

Version 2

Referee Report 13 September 2016

doi:10.5256/f1000research.9391.r15427

Christian Cole
Division of Computational Biology, School of Life Sciences, University of Dundee, Dundee, UK

Daouda *et al.* propose a new tool, pyGeno, for the interrogation of proteomics data in the context of genomic sequence variants.

General Comment:

- Several times the authors make reference to 'precision medicine' without clarifying what is meant by the term. If the authors have a specific workflow or use case which befits 'precision medicine' they need to make it clearer. I note that the authors do not include a 'Use Cases' section as suggested by in the 'Instructions to Authors'. Their previous paper (Granados *et al*, 2014) would be a great example of how to use pyGeno on real data.The examples provided in the paper are either too vague or too simplistic.

- Too often new tools come out which try to reinvent the wheel for a small incremental improvement. Here, the authors need to be acknowledged for using well-established systems like Ensembl, SQLite and reading in existing data types (e.g. GFF, VCF, fasta).

- The paper is too short on specifics and somewhat unstructured. The Methods section is fine although would benefit from an overview- with a Figure and/ or text- as the overall structure of pyGeno is unclear. The Personalized genomes section should be expanded as a 'Use Cases' section. The 'Operation' section should be part of the Methods.

- The online instructions are quite clear for installation, however I gave up due to the incredibly slow progress: >20 hours remaining of a full genome 'datawrap'. Thus, unfortunately, I was not able to test the software myself. Given that pyGeno has been downloaded many times, this might an issue local to me.

- Overall I do feel pyGeno is a valuable contribution to the community, however the paper needs some improvement to highlight the tool's usefulness better.

Specific Comments:

- In the third paragraph of the Methods, the authors state that pyGeno is not dependent on any 'remote REST APIs'. If this is the case how does the pyGeno interact with Ensembl and keep in sync with the regular updates (every 6 months)? The focus on 'robust and faster processing' is

understandable, but version drift from official sources can be a serious problem. Is version maintenance something end users can do or are they dependent on the authors keeping the versions up-to-date?

- In Figure 1, a simple example is provided showing a protein sequence with what appears to be two non-synonymous variants highlighted in the protein sequence. How does pyGeno cope with summarising/ visualising; 1) mutually exclusive variants at a single amino acid (e.g. two non-synonymous variants at different positions of the codon), or 2) more complex variants like splicing-affecting changes and loss/gain of STOP codons? Similarly does pyGeno accept phased haplotypes thereby allowing inspection of both protein products from each of the individual's alleles?

- What are the hardware requirements for running pyGeno and associated analyses? Is a well-specified workstation with several GB of RAM, fast cpu and terabytes of diskspace required or can it be run on a laptop?

- The final paragraph does not contribute anything to the paper, I suggest the authors remove it and end the article with something more succinct and pertinent.

References

1. Granados DP, Sriranganadane D, Daouda T, Zieger A, Laumont CM, Caron-Lizotte O, Boucher G, Hardy MP, Gendron P, Côté C, Lemieux S, Thibault P, Perreault C: Impact of genomic polymorphisms on the repertoire of human MHC class I-associated peptides.*Nat Commun*. 2014; **5**: 3600 PubMed Abstract | Publisher Full Text

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Competing Interests: No competing interests were disclosed.

Referee Report 31 May 2016

doi:10.5256/f1000research.9391.r14063

Hilary Coller
Department of Molecular, Cell and Developmental Biology, University of California Los Angeles, Los Angeles, CA, USA

This is manuscript describes the development of a valuable new Python package that provides the user with an environment in which they can explore multiple different large datasets related to a single gene. The software maps genes back to Ensemble, provides polymorphisms from dbSN, and provides information on experimentally detected patient-specific polymorphisms. The software can also handle DNA Sequencing data, RNA-Seq data and proteomics data for the same individual. The authors have designed the software so that it manages these datasets efficiently, thus providing the user with seamless and rapid access to the information desired. The use of "datawraps" so that co-authors can share information will likely also be valuable for users. The software has been downloaded over 12,000 times in the first year, demonstrating its utility. There is a very helpful figure, Figure 1, that gives an overview of the

process. The authors might consider adding a section of the manuscript in which they walk the reader through the analysis of an actual gene in an actual genome to give the reader a sense of the findings.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Referee Report 31 May 2016

doi:10.5256/f1000research.9391.r13874

Lynn Fink
Diamantina Institute, University of Queensland, Brisbane, Australia

pyGeno is a Python package that allows a user to simply query either a standard reference genome or a custom genome for information such as sequences, SNPs, and related RNA and protein data. I agree that this package fills a need for genome research, appears to be very straightforward to use, and I would like to use it myself.

However, when I tested it I was unable to get it to run; contrary to the authors claims that it is easy to install with minimal dependencies I couldn't make it work. (I am not a Python expert, but I have installed much more complicated packages successfully.) With Python virtual environments, I wonder if there is a need to be quite so minimal. Would it be easier to rely on standard packages to ensure a universal, smooth experience?

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Competing Interests: No competing interests were disclosed.

Author Response 03 Jun 2016

Tariq Daouda, University of Montreal, Canada

Thank you for taking the time to review our work.

In light of your review we have retested the installation, genome importation and polymorphism insertion on Linux, MacOS and Windows and it seems to work on every platform. We are fully committed to make the experience for the end user as smooth and easy as possible, if you could give us more details about the problems you encountered by filling a GitHub issue, we would be very happy to address them.

Competing Interests: No competing interests were disclosed.