Guha *J Cheminform* (2019) 11:12
https://doi.org/10.1186/s13321-019-0333-z

Journal of Cheminformatics

EDITORIAL Open Access

Implementing cheminformatics

Rajarshi Guha*

Content

Computational characterization of chemical structures originated before the advent of digital computers [1]. However, the ability to represent and manipulate large collections of molecules and their associated information was enabled by the rise of cheminformatics algorithms and their implementions on digital computers. Willett [2] has suggested the work of Ray and Kirsch [3] on substructure searching as the first description of a computer implementation (on punched cards) of a cheminformatics algorithm.

Programming language research blossomed during the 1950's and 60's and saw the development of high level programming languages (such as FORTRAN [4], LISP [5] and ALGOL [6]). Cheminformatics research took advantage of these efforts, to move beyond punched cards. One of the earliest cheminformatics applications in a high level language was DENDRAL [7], written in LISP in 1963 [8]

Since the 1960's, a plethora of languages have come into existence. Each language has its distinct features (directly memory manipulation in C, code as data in LISP [9], automated memory management in Java, lazy evaluation [10] in Haskell), but useful features from one language tend to show up in others (e.g., automated memory management initially appeared in LISP, but is now found in Java, Ruby, Python, C# and others). Furthermore, all modern languages are Turing equivalent [11] (i.e., capable of performing any arbitrary computation). One might then ask, what does it matter what language one uses to implement cheminformatics?

A number of factors go into deciding what language to use in a given setting. These include the suitability for a specific task (web development versus statistical modeling), prior knowledge of the language, the availability of supporting tools & frameworks and their licensing requirements and of course, performance.

A key consideration is the availability of external libraries such as cheminformatics toolkits (e.g., CDK [12] or JChem for Java applications). Many libraries (especially those written in C or C++) can be wrapped and made accessible to other languages (e.g., OpenBabel [13], RDKit and OEChem which are written in C++ provide SWIG wrappers enabling their use in Python and Java). Finally, for many projects, the choice of language is dictated by historical development (such as the use of Fortran for much of scientific computing).

At a more fundamental level, there are different programming models, which require conceptually different approaches to designing an application. For example, Khomtchouk et al. [14] suggest that the functional paradigm is best suited for scientific software development. On the other hand, Ray et al. [15] show that projects using functional languages do not necessarily show better software quality. One must consider others aspects, ranging from performance issues to the availability of programmers with sufficient skills to develop and then maintain applications written in functional languages. It is useful to note that some languages such as Scala are a hybrid, supporting both functional and procedural paradigms.

In this thematic series we have invited authors to present their views on a variety of programming languages. The series is rolling, and starts of with contributions from Thiesen [16], Berenger [17], and Höck [18] discussing JavaScript, OCaml and Scala respectively. We anticipate contributions covering Scala, C/C++, Tcl and noSQL.

The intended audience for this series are practitioners of cheminformatics who are already familiar with one programming language and would like to learn what other languages may offer in terms of language features and supported tooling.

*Correspondence: rajarshi_guha@vrtx.com
Vertex Pharmaceuticals, 50 Northern Ave, Boston, MA 02210, USA

© The Author(s) 2019. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.

We do not intend this to be a head to head comparison. Rather, the contributions are structured to address one or more of the following aspects

- How that language (or model of programming) affects scientific software development
- How a language may enable the development of new approaches to solving a problem in cheminformatics or computational chemistry
- Specific approaches to overcome language limitations when dealing with chemical of biological data types
- Comments on performance and it's relevance to the languages goals
- Educational aspects of the language (is it easier for newcomers?)
- Development environments and frameworks that make a language easier to use and deploy (e.g., RStudio for R and Jupyter notebooks for Python)

The goal of this issue is to highlight features of different languages that the authors have employed to build applications as well as their views on the benefits (and downsides) of the language that has driven them to invest effort in building capabilities in their chosen language. We do not expect that this will identify any single language as the "chosen one". Rather, we hope that the articles in this issue will be a useful guide for the community to assess which languages may be appropriate for their next project.

Author's contributions
RG conceived and designed the thematic issue and wrote this manuscript. The author read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 6 December 2018 Accepted: 14 January 2019
Published online: 05 February 2019

References
1. Wiener H (1947) Structural determination of paraffin boiling points. J Am Chem Soc 69(11):2636–2638
2. Willett P (2011) Chemoinformatics: a history. WIREs Comput Mol Sci 1(1):46–56
3. Ray LC, Kirsch RA (1957) Finding chemical records by digital computers. Science 126:814–819
4. McJones P (2018) History of FORTRAN and FORTRAN II (2018). http://www.softwarepreservation.org/projects/FORTRAN Accessed Nov 2018
5. Stoyan H (1984) Early lisp history (1956–1959). In: Proceedings of the 1984 ACM symposium on LISP and functional programming. LFP '84, pp 299–310. ACM, New York. https://doi.org/10.1145/800055.802047
6. McJones P (2018) History of ALGOL (2018). http://www.softwarepreservation.org/projects/ALGOL/. Accessed Nov 2018
7. Lindsay RK, Buchanan BG, Feigenbaum EA, Lederberg JA (1993) DENDRAL: a case study of the first expert system for scientific hypothesis formation. Artif Intell 61(2):209–261
8. Sutherland G (1963) Letter from Georgia Sutherland to R. Shirley. https://exhibits.stanford.edu/feigenbaum/catalog/qc171fk5406
9. McIlroy D (1960) Macro instruction extensions of compiler languages. Commun ACM 3(4):214–220
10. Watt DA, Findlay W (2004) Programming language design concepts. Wiley, Hoboken
11. Brainerd WS, Landweber LH (1974) Theory of computation. Wiley, Hoboken
12. Willighagen EL, May JW, Alvarsson J, Berg A, Carlsson L, Duhrkop K, Jeliazkova N, Kuhn S, Pluskal T, Rojas-Cherto M, Spjuth O, Torrance G, Evelo CT, Guha R, Steinbeck C (2017) The chemistry development kit (cdk): atom typing, rendering, molecular formulas, and substructure searching. J Cheminform 9:33
13. O'Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR (2011) Open babel: an open chemical toolbox. J Cheminform 3:33. https://doi.org/10.1186/1758-2946-3-33
14. Khomtchouk BB, Weitz E, Karp PD, Wahlestedt C (2018) How the strengths of lisp-family languages facilitate building complex and flexible bioinformatics applications. Brief Bioinform 19(3):537–543. https://doi.org/10.1093/bib/bbw130
15. Ray B, Posnett D, Devanbu P, Filkov V (2017) A large-scale study of programming languages and code quality in github. Commun ACM 60(10):91–100
16. Theisen KJ (2019) Programming languages in chemistry: a review of HTML5/JavaScript. J Cheminform. https://doi.org/10.1186/s13321-019-0331-1
17. Berenger F, Zhang KYJ, Yamanishi Y (2019) Chemoinformatics and structural bioinformatics in OCaml. J Cheminform. https://doi.org/10.1186/s13321-019-0332-0
18. Höck S, Riedl R (2012) chemf: a purely functional chemistry toolkit. J Cheminform 4(1):38. https://doi.org/10.1186/1758-2946-4-38

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

BMC