BIOINFORMATICS

Vol. 00 no. 00 2014
Pages 1–2

DNAplotlib: standardized visualization of genetic constructs, libraries and associated data

Thomas E. Gorochowski[1], Bryan Der[1], Emerson Glassey[1], D. Benjamin Gordon[1] and Christopher A. Voigt[1,*]

[1]Department of Biological Engineering, Synthetic Biology Center, Massachusetts Institute of Technology, USA.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Summary: DNAplotlib is a computational toolkit that enables highly customizable visualization of single genetic constructs and libraries of design variants. Publication quality vector-based output is produced and all aspects of the rendering process can be easily customized or extended by the user. DNAplotlib is capable of outputting SBOL Visual compliant diagrams, in addition to a trace-based format that is able to better illustrate the precise location and length of each genetic part. This alternative visualization method enables direct comparison with nucleotide-level data such as RNA-seq read depth. While it is envisaged that access will be predominantly via the programming interface, command-line and web-based front-ends are also provided to support broader usage.

Availability: DNAplotlib is cross-platform and open-source software developed using Python and released under the OSI recognized NPOSL-3.0 licence. Source code, documentation and a web front-end are available at the project website: http://www.dnaplotlib.org.

Contact: cavoigt@gmail.com

Engineering disciplines rely on standardized pictorial representations of parts and their interconnections to clearly communicate how these should be pieced together to allow for the reliable construction of large complex systems. In biology, DNA sequences are often engineered to create genetic constructs that probe or perturb the function of natural systems, or more recently, create novel capabilities in what has been termed "synthetic biology" (Church *et al*., 2014). Unlike traditional engineering fields, the way that these designs are visually represented varies significantly between labs and in different areas of the field, leading to ambiguities that can hinder understanding and the effective reuse of this research. The Synthetic Biology Open Language (SBOL) Visual initiative was started to help alleviate this problem by defining a set of agreed symbols for commonly used genetic elements (Quinn *et al*., 2013). However, so far this standard has seen limited uptake due to a lack of accessible tools that can be directly integrated into existing design and analysis workflows.

Some attempts to automate the creation of standard-compliant diagrams have been made, with the most prominent being PigeonCAD (Bhatia & Densmore, 2013). This web-based tool interprets a custom text-based syntax used to specify a genetic design and automatically transforms this into visual representation. The major problem with this approach is that the syntax provides limited ability to customise the visualisations produced, the output is

Furthermore,

While this tool is Unfortunately, this tool is hampered by several limitations. The biggest limitation of these approaches so far though is an inability to integrate with existing scientific computing visualization and analysis tools. This is essential as

Another growing area of research that is not well supported by such existing tools is the push towards automated design procedures that harness the potential to construct huge libraries of design variants (Smanski *et al*., 2014; Bilitchenko *et al*., 2011). Under these scenarios, automation of visualization tasks is essential to ensure clear communication of these large and diverse design spaces. No existing tools are available to support this need.

As synthetic biology moves ever closer to automated design procedures that harness the potential to construct huge libraries of design variants (Smanski *et al*., 2014; Bilitchenko *et al*., 2011), supporting visualization tools will become important to ensure clear communication of these large and diverse designs.

To tackle these limitations we developed DNAplotlib, a computational toolkit that facilitates highly-customisable visualiations of genetic constucts. By providing access at the levels

DNAplotlib has been built using matplotlib (Hunter, 2007), a Python-based 2D graphics library that allows for graphical output in the form of vector-based PDFs or rasterized images. Python was chosen as the underlying language due to its increasing use during the analysis of biological data (Cock *et al*., 2009), which ensured that visualizations generated by DNAplotlib could be easily integrated into existing analysis scripts and workflows with minimal effort. Furthermore, Python is highly-portable ensuring availability across all major operating systems.

Designs are stored as a standard Python list where each element is a dictionary defining details regarding the part at that position in the construct. Visualisations are generated automatically by scanning this data structure and calling functions associated to the type of part at each position. Standard functions are provided to generate SBOL Visual symbols as well as

Regulation is handled in a similar way with arcs.

*to whom correspondence should be addressed

A key consideration in the design of DNAplotlib was ensuring that all aspects of the rendering process can be customized to a users' requirements. The rapid pace at which biological research progresses has resulted in the continual discovery of new forms of genetic part.

As fields such as synthetic biology are still evolving, the the precise way in which such tools will be used and the discovery of new types of genetic part that need to be included, but may not yet have a standardized representation. Enabling custom elements to be easily added and refined over time will ensure such elements are captured and also contribute to the standardization process.

dictionary that contains core attributes corresponding to part type and orientation, in addition to other attributes that influence the style of the part. Attributes that are not used by a particular renderer are ignored, meaning that it is easy to swap these functions without fear of breaking the entire rendering pipeline.

Although directly accessing DNAplotlib using Python gives the greatest flexibility, we recognized that in many cases

To generate a In addition to direct library access through Python analysis scripts, we also provide two scripts to enable input in the form of text files

To ensure broadest application of these tools by non-programmers, a web-based interface is also provided. The website is built using Jetty and enables a user to simply upload

This dire access to the two scripts described previously and removes the need for a user to have a local installation of Python or the library. The websites run using on top of the Jetty ...

DNAplotlib is under continual development with a current focus on broadening the types of genetic element covered to include new synthetic biological parts. The project welcomes contributions from others within the community through the project website and public development repository: http://www.dnaplotlib.org.

ACKNOWLEDGEMENTS

T.E.G., B.D., E.G., D.B.G. and C.A.V. were supported by...

REFERENCES

Church, G.M., Elowitz, M.B., Smolke, C.D., Voigt, C.A. and Weiss, R. (2014). Realizing the potential of synthetic biology, *Nat. Rev. Mol. Cell Biol.*, **15**, 289-294.

Bhatia, S. and Densmore, D. (2013). Pigeon: A Design Visualizer for Synthetic Biology, *ACS Synth. Biol.*, **2**, 348-350.

Smanski, M.J., Swapnil, B., Park, YJ., Zhao, D., Giannoukos, G., Ciulla, D., Busby, M., Calderon, J., Nicol, R., Gordon, D.B., Densmore, D. and Voigt, C.A. (2014) Combinatorial design and assembly of refactored gene clusters, *Nat. Biotech.*, **?**, ???-???.

Hunter, J.D. (2007). Matplotlib: A 2D graphics environment, *Computing in Science & Engineering*, **9**, 90-95.

Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, and de Hoon MJ. (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422-1422.

Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J.C., Densmore, D. (2011) Eugene A Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems. *PLoS ONE*, **6**, e18882.

Quinn, J., Beal, J., Bhatia, S., Cai, P., Chen, J., Clancy, K., Hillson, N., Galdzicki, M., Maheshwari, A.P., Umesh; P., Matthew; R.C.; Stan, G.-B., Endy, D. (2013) "Synthetic Biology Open Language Visual (SBOL Visual), version 1.0.0."

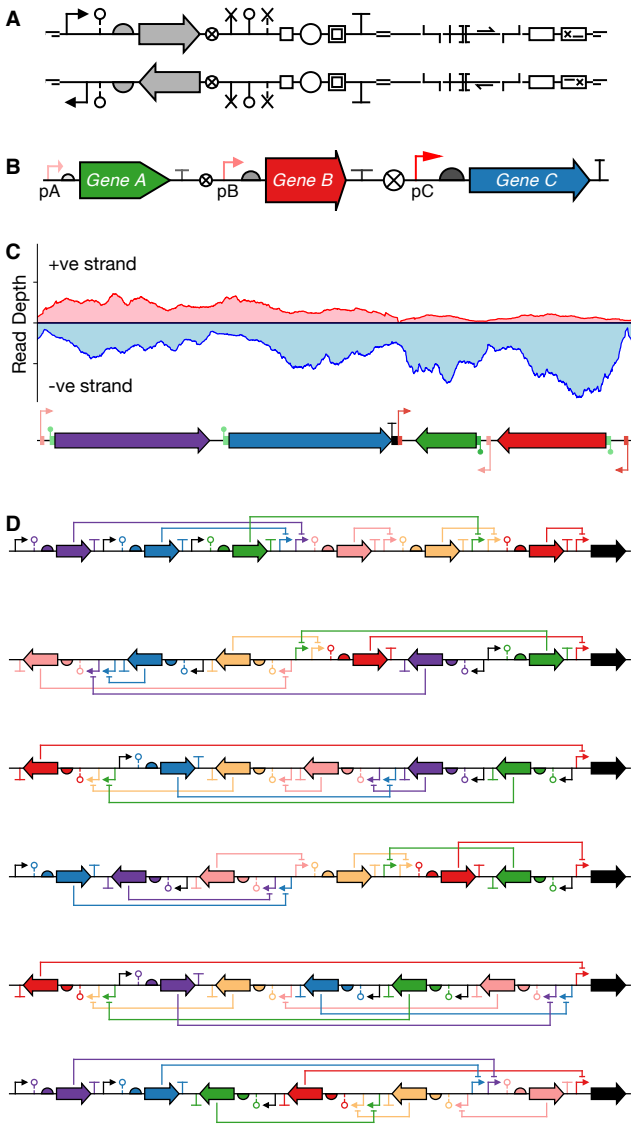

Fig. 1. Overview of core DNAplotlib functionality. All visualizations were generated driectly from DNAplotlib. (**A**) The complete set of SBOL Visual parts that capture the majority of widely used genetic elements are available for use in both forward and reverse orientations. (**B**) The size, color, shape and labeling of all elements can be easily customized allowing for additional information to be communicated e.g., promoter strengths or spacer lengths. Users can further supply their own functions to draw parts in a non-standard way or to represent new types of genetic element yet to be incorporated into SBOL Visual. (**C**) To allow for direct comparison between a genetic design and associated nucleotide-level data, such as RNA-seq read depths, trace-based renderers are also provided. These use the standard promoter and terminator symbols and a small filled circle to represent an RBS. Indicators of actual part widths (arrow length for coding sequences and small filled rectangles for promoters and RBSs) are displayed on the DNA backbone to enable a clear visual alignment of design information with trace data. (**D**) Visualization of a library of genetic design variants implementing the same 3-input (black promoters), 1-output (black coding sequences) device. Colors have been used to link repressor genes to their cognate promoters.