Vol. 00 no. 00 2014
Pages 1–2

DNAplotlib: programmable 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 license. 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 and 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 genetic designs are visually represented varies significantly between labs and across different areas of the field. This leads to ambiguities that can hinder understanding and the effective reuse of 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), see Fig. 1A. 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.

To our knowledge, the only attempt so far to automate the creation of standard-compliant diagrams has been Pigeon (Bhatia & Densmore, 2013). This web-based tool interprets a custom syntax used to specify genetic designs and automatically transforms these into visual representations. While Pigeon greatly simplifies the creation of visual genetic designs, it was not developed with the goal of giving users direct access to the rendering procedure itself. This is not a problem in many cases, but has meant that Pigeon is not able to accommodate more sophisticated programmable customizations that require alterations to the symbol shapes, color or labeling, and has led to only a partial set of SBOL symbols being available due to the continual evolution of the the SBOL Visual standard. Furthermore, synthetic biology is increasingly moving towards automated design procedures that have the potential to construct huge libraries of design variants (Smanski *et al.*, 2014; Bilitchenko *et al.*, 2011), see Fig. 1D. Under these scenarios, efficient automation of visualization tasks is essential to ensure clear communication of these large design spaces. Pigeon was not designed for direct integration into other workflows making it unsuitable for such tasks.

To address these limitations we developed a computational toolkit called DNAplotlib that enables highly-customizable visualization of genetic constructs in a programmable way (Fig. 1B). DNAplotlib has been developed in Python and makes significant use of matplotlib (Hunter, 2007), a 2D graphics library that can produce graphical output in the form of vector-based PDFs or rasterized images. Python was chosen as the underlying language due to its increasing use in the analysis of biological data (Cock *et al.*, 2009). This enables visualizations generated by DNAplotlib to be integrated into existing analysis workflows with minimal effort. Furthermore, Python is highly-portable ensuring availability of our tools across all major operating systems.

At the core of the toolkit is the main rendering pipeline. This is structured such that individual functions are provided when run to draw each symbol type. Standard built-in functions can be chosen that cover the full range of SBOL Visual parts (Fig. 1A), or the user can specify their own which may include new forms of part type not currently available. In addition, we also provide what are termed trace-based renderers that draw promoters, ribosome binding sites (RBSs), coding sequences (CDSs) and terminators such that the start and end points relate to their actual position within a construct (Fig. 1C). Unlike the conceptual SBOL Visual renderers where only ordering and orientation information are maintained, trace-based

*to whom correspondence should be addressed

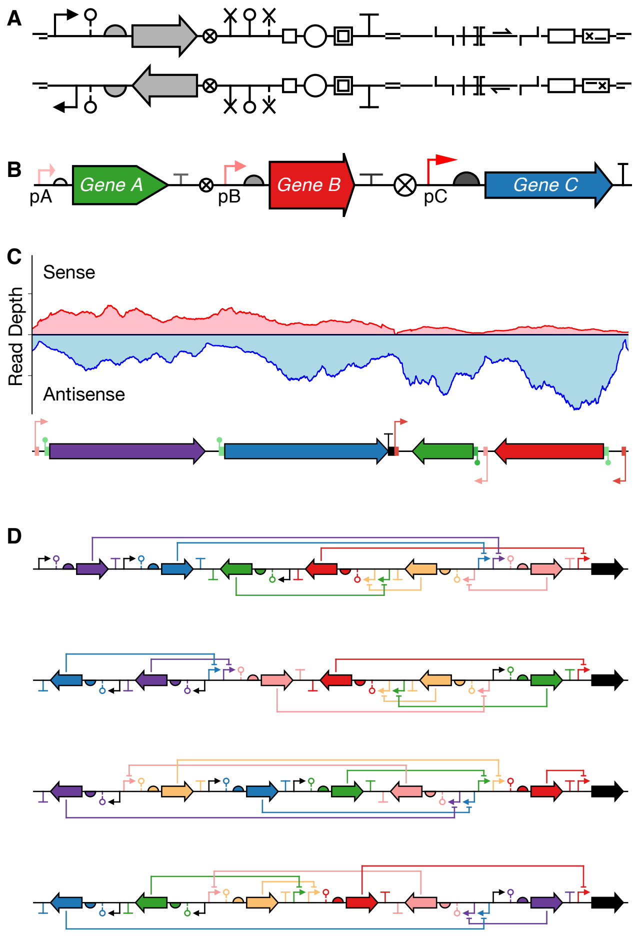

Fig. 1. Overview of core DNAplotlib functionality. All visualizations were generated driectly from DNAplotlib. (**A**) The complete set of SBOL Visual parts that capture the majority of widely used genetic elements are available in both forward and reverse orientations. (**B**) The size, color, shape and labeling of all elements can be easily customized enabling 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 facilitate direct comparison between a genetic design and associated nucleotide-level data (e.g., 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.

renders allow for a direct comparison between nucleotide-level data and the associated design information, see Fig. 1C.

To create a visualization, designs are provided in the form of a standard list data type, where each element is a dictionary defining the part at that position and any other design information, e.g., orientation, length and styling options. By using a dictionary, varying options for each part type can be easily accommodated. Visualizations are then generated by scanning this list and calling functions associated to each part type encountered. If a part type is unknown or an attribute not used, this element is ignored ensuring that renderers providing differing levels of functionality do not break the entire pipeline. Regulation is handled in a similar way with start and end points provided in addition to styling options. All rendering is performed using a matplotlib axis, allowing for genetic designs to be incorporated into standard plotting routines.

Although directly accessing DNAplotlib from Python gives greatest flexibility, in many cases it is simpler for a user to specify designs and part customizations in text-based files. These can be shared more easily and allow for better reuse of design or styling information. For these purposes we developed two command-line interfaces. The first mimics the idea of Pigeon using a simple syntax to define basic constructs as a single line of text. This is useful for the quick creation of small constructs with limited customization. The second is designed for the visualization of large libraries. Users are required to provide several text files defining the parts, their styling, and the part ordering, orientation and regulatory links. These are then processed and a visualization of the full library of designs is generated.

To further ensure broadest access to non-programmers, web-based interfaces for each of these command-line scripts is available. These were developed using Jetty and provide a graphical user interface for creating simple constructs and the ability to easily upload and process text-based files defining a library of designs. Visualizations can be previewed and downloaded as JPEG or PDF files.

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.”