BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 5 2006, pages 616–617
doi:10.1093/bioinformatics/btk021

Genome analysis

GenomeDiagram: a python package for the visualization of large-scale genomic data

Leighton Pritchard, Jennifer A. White, Paul R.J. Birch and Ian K. Toth
Plant Pathogen Programme, Scottish Crop Research Institute, Invergowrie, Dundee, DD2 5DA, Scotland, UK

Received on October 27, 2005; revised on December 8, 2005; accepted on December 21, 2005
Advance Access publication December 23, 2005
Associate Editor: Martin Bishop

ABSTRACT

Summary: We present GenomeDiagram, a flexible, open-source Python module for the visualization of large-scale genomic, comparative genomic and other data with reference to a single chromosome or other biological sequence. GenomeDiagram may be used to generate publication-quality vector graphics, rastered images and in-line streamed graphics for webpages. The package integrates with datatypes from the BioPython project, and is available for Windows, Linux and Mac OS X systems.

Availability: GenomeDiagram is freely available as source code (under GNU Public License) at http://bioinf.scri.ac.uk/lp/programs.html, and requires Python 2.3 or higher, and recent versions of the ReportLab and BioPython packages.

Contact: lpritc@scri.ac.uk

Supplementary Information A user manual, example code and images are available at http://bioinf.scri.ac.uk/lp/programs.html

1 INTRODUCTION

High-throughput genome sequencing and analyses now routinely generate vast quantities of data, permitting comparisons of whole genome structure and organisation, and gene expression studies across multiple complete genomes. The interpretation and understanding of relationships revealed by these data is greatly improved if they can be visualized in a simple manner.

GenomeDiagram is a module written for the powerful Python high-level programming language, providing a straightforward interface for the generation of publication-quality vector, raster and streamed images. The images constructed by the package can represent very large amounts of biological information, ordered in relation to a reference sequence. The package integrates with the freely-available BioPython bioinformatics libraries for Python, and the ReportLab backend for rendering images.

The GenomeDiagram package was designed for the display of large-scale comparative genomics data (Fig. 1), allowing the identification of sequence similarities and insertions across hundreds of genomic sequences simultaneously. GenomeDiagram has also found use in the display of results from time-course microarray experiments, with probes located on the genome of interest, permitting the identification of co-regulated, co-localized genes.

*To whom correspondence should be addressed.

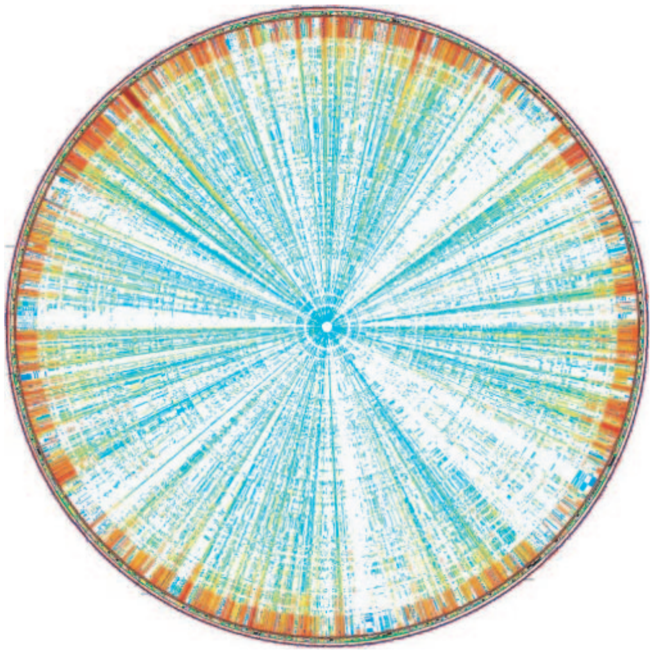

Fig. 1. Circular diagram rendered by GenomeDiagram of the reciprocal best hit comparison of all coding sequences from *Eca* against 229 bacterial genomes. Individual hits are coloured in a graduated scheme from dark red to light blue in order of decreasing sequence similarity. Successive inner rings represent distinct bacterial genomes in order of decreasing average similarity of coding sequence. The image summarizes over 1 000 000 data points.

2 FEATURES AND BASIC USAGE

The GenomeDiagram package may be used to construct circular and linear diagrams in which information is presented on a series of ‘tracks’ or levels. Each ‘track’ may hold information about ‘features’ or graphs of data using the location on the reference sequence as the abscissa. Features may be present or absent for any point on the reference sequence, and if present may also be individually coloured to represent a data value or some other classification. Graphs may be drawn in line, bar or heat graph format.

GenomeDiagram provides a simple API in Python for the generation and manipulation of these diagrams (Table 1), and integrates with the open source projects BioPython

Table 1. GenomeDiagram example code

```
# Import Python modules
from Bio import GenBank
from reportlab.lib import colors
from GenomeDiagram import GDDiagram, GDUtilities
# Load genome annotations from GenBank file
parser = GenBank.FeatureParser()
fhandle = open('NC_005085.gbk','r')
genbank_entry = parser.parse(fhandle)
fhandle.close()
# Draw linear diagram of CDS features, with GC content graph
gdd = GDDiagram('NC_005085.gbk')
gdt1 = gdd.new_track(4, greytrack=1, name='Cv CDS',
   scale_fontsize=3, greytrack_fontsize=3)
gdt2 = gdd.new_track(6, greytrack=1, name='Cv GC content',
   scale_fontsize=3, greytrack_fontsize=3, height=2)
gdfs = gdt1.new_set('feature')
gdgs = gdt2.new_set('graph')
graphdata1 = GDUtilities.gc_content(genbank_entry.seq, 1000)
graph1 = gdgs.new_graph(graphdata1, 'GC content', style='line',
   colour=colors.blue, altcolour=colors.purple)
graph1.linewidth=1
for feature in genbank_entry.features:
   if feature.type == 'CDS':
      gdfs.add_feature(feature, colour=colors.red)
gdd.draw(format='linear', orientation='landscape', tracklines=0,
   pagesize='A6', fragments=10, circular=1)
# Write image as a PNG raster file, and as a PDF vector image
gdd.write('example1.png', 'PNG')
gdd.write('example1.pdf', 'PDF')
```

(http://www.biopython.org—for biological sequence manipulation) and ReportLab (http://www.reportlab.org—for rendering images).

Images of arbitrary size may be generated using GenomeDiagram in scalable vector formats such as Adobe PDF, PostScript (PS), Encapsulated PostScript (EPS) and Scalable Vector Graphic (SVG), as well as in common raster formats such as JPG, BMP, TIF and PNG. The package can be used to write the images to a static file or stream the image (e.g. as part of a .cgi application).

3 DISCUSSION

GenomeDiagram was used to visualize comparative genomics studies of the plant pathogen *Erwinia carotovora* subsp. *atroseptica* (*Eca*) (Fig. 1). Regions of potential horizontal gene transfer within the *Eca* genome, and other regions including the *cfa* cluster encoding synthetic proteins for a phytotoxin, and a cluster of nitrogen-fixation genes, associated with pathogenicity and its plant-associated lifestyle, were readily identified and visualized with this package. (Bell *et al*., 2004). The GenomeDiagram package has since also been used to visualize time-course microarray data, indicating co-regulated and co-localized genes on a reference genome, and to stream genome maps and context diagrams for web use.

GenomeDiagram generates static images of genomes and associated data, such as genome comparisons, as do GenomeViz (Ghai *et al*., 2004), GeneWiz (Jensen *et al*., 1999) and EnteriX (Florea *et al*., 2003), which generate PostScript or PDF format images at restricted sizes. GenomeDiagram can also generate these vector images, but may also additionally generate bitmaps in several formats, at arbitrary page sizes.

Unlike interactive viewers such as ACT (Carver *et al*., 2005), WebACT (Abbott *et al*., 2005), MAUVE (Darling *et al*., 2004) and the MGV linear sequence viewer (Kerkhoven *et al*., 2004), GenomeDiagram output does not allow 'mouseover' or 'click-through' interaction by the user although it can, like MGV, provide SVG format output that may contain such interactive elements. These applications, like GenomeViz, GeneWiz and Enterix, face computational and design restrictions on the number of sequences (and in some cases, the sequences themselves) that may simultaneously be viewed. In contrast, GenomeDiagram is able to display an arbitrary number of user-defined sequences and features on a single diagram (Fig. 1).

The MAUVE, ACT, WebACT and EnteriX viewers are restricted to linear and GenomeViz to circular diagram formats. MGV and GeneWiz are capable of rendering both the diagram types, but linear diagrams only show a section of the genome. In all cases, input data formats are either non-standard (e.g. GenomeViz MAP files), restricted to a small subset of standard formats or the input data are not user accessible. GenomeDiagram both diagrams and may represent any data that can be a BioPython SeqFeature object. Parsers for standard input formats are available in BioPython.

GenomeDiagram is a programming library, designed to provide flexibility in diagram design and automation while still providing a simple and accessible API that can be used from other applications. The flexibility and potential for automation of GenomeDiagram is not available in the standalone and online viewers discussed above.

A graphical program for the generation of publication-quality genome circular diagrams on Windows, built upon GenomeDiagram, is available at http://bioinf.scri.ac.uk/lp/programs.html, and a cross-platform graphical user interface implemented using wxPython (www.wxpython.org), is also in development.

ACKNOWLEDGEMENTS

We would like to thank Robin Becker for help with ReportLab and Mylnefield Research Services for financial support.

Conflict of Interest: none declared.

REFERENCES

Abbott,J.C. *et al*. (2005) WebACT — an online companion for the Artemis Comparison Tool. *Bioinformatics*, **21**, 3665–3666.

Bell,K. *et al*. (2004) Genome sequence of the enterobacterial phytopathogen *Erwinia carotovora* subsp. *atroseptica* and characterization of virulence factors. *Proc. Natl Acad. Sci. USA*, **101**, 11105–11110.

Carver,T.J. *et al*. (2005) ACT: the Artemis Comparison Tool. *Bioinformatics*, **21**, 3422–3423.

Darling,A.C. *et al*. (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res*., **14**, 1394–1403.

Florea,L. *et al*. (2003) EnteriX 2003: Visualization tools for genome alignments of Enterobacteriaceae. *Nucleic Acids Res*., **31**, 3527–3532.

Ghai,R. *et al*. (2004) GenomeViz: visualizing microbial genomics. *BMC Bioinformatics*, **5**, 198.

Jensen,L.J. *et al*. (1999) Three views of microbial genomes. *Res. Microbiol*., **150**, 773–777.

Kerkhoven,R. *et al*. (2004) Visualization for genomics: the Microbial Genome Viewer. *Bioinformatics*, **20**, 1812–1814.