BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 20 2013, pages 2655–2656
doi:10.1093/bioinformatics/btt448

Data and text mining

Advance Access publication August 5, 2013

Scaffold network generator: a tool for mining molecular structures

Matt K. Matlock, Jed M. Zaretzki and S. Joshua Swamidass*

Department of Pathology and Immunology, Washington University School of Medicine, Campus Box 1097, Whitaker Hall, St. Louis, MO 63130, USA

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: Scaffold network generator (SNG) is an open-source command-line utility that computes the hierarchical network of scaffolds that define a large set of input molecules. Scaffold networks are useful for visualizing, analysing and understanding the chemical data that is increasingly available through large public repositories like PubChem. For example, some groups have used scaffold networks to identify missed-actives in high-throughput screens of small molecules with bioassays. Substantially improving on existing software, SNG is robust enough to work on millions of molecules at a time with a simple command-line interface.

Availability and implementation: SNG is accessible at http://swami.wustl.edu/sng

Contact: swamidass@wustl.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on March 11, 2013; revised on July 22, 2013; accepted on July 31, 2013

1 INTRODUCTION

Identifying the molecular scaffolds—the core structure of a molecule with its side-chains are removed—that are biologically active is a central task in drug discovery. Drugs are discovered by chemically modifying these molecular scaffolds to improve their medicinal properties (Proudfoot, 2002). Viewing molecules through the lens of scaffolds is informative. For example, Bemis and Murcko (1996) found that a majority of FDA approved drugs are derived from a small set of scaffolds. Similarly, Swamidass *et al*. (2012) found that maximizing the diversity of scaffolds selected for confirmatory testing can dramatically improve the efficiency of screening experiments.

Detecting and defining scaffolds, however, is subjective because there is ambiguity in which part of a molecule's structure should be considered the core scaffold and which parts should be considered side-chains. Early work in the field ignores these ambiguities and uses an algorithm to assign each molecule to a single scaffold that may not be ideal. For example, the Bemis–Murko scaffold of Invacaftor is shown at the top level in Figure 1.

Subsequent work more appropriately captured the complexity of extracting scaffolds by assigning molecules to a node in a network or tree of interrelated scaffolds. The first strategy for managing scaffold ambiguity is the Scaffold Tree (Schuffenhauer *et al*., 2007). This algorithm assigns each molecule to a single Bemis–Murcko scaffold, but then decomposes this scaffold by iteratively stripping the most peripheral ring. Successive decompositions yield a linear graph of scaffolds, all of which are assigned to the molecule. For example, the linear graph generated from Invacaftor depicted with double-dotted lines in Figure 1. A Scaffold Tree is the aggregation of the linear graphs derived from multiple molecules. Scaffold trees are somewhat helpful, but introduce a new ambiguity: defining which ring is the most 'peripheral.'

The second strategy, Scaffold Networks, decomposes each scaffold in all possible ways. This creates a network of scaffolds for each molecule. The scaffold network for Invacaftor is shown in Figure 1. Networks from multiple molecules can be superimposed to form a large multipartite graph (Varin *et al*., 2011). Molecules associated with nodes in this graph—each of which correspond to specific scaffolds—often have similar properties. This network of scaffolds is useful for analysing and visualizing screening data to pick out the most promising scaffolds to develop into drugs.

Until now, no open-source software could generate both Scaffold Trees and Networks for large sets (>225 000) of molecules. We fill this gap by releasing a command-line utility that extracts scaffold networks for large datasets and outputs them in an easy to parse format that facilitates computational analysis.

2 SCAFFOLD NETWORK GENERATOR

Scaffold network generator (SNG) is an open-source command-line utility that computes the hierarchical network of scaffolds for a given set of input molecules for use in an automated drug discovery pipeline. SNG outputs either scaffold trees or scaffold networks and can process datasets containing over 10 million molecules in under 24 h.

There are several proprietary and open-source systems that compute Bemis–Murcko scaffolds. Just two of these tools are open source—the GUI-based application *Scaffold Hunter* (Wetzel *et al*., 2009) (SH) and its antiquated command-line predecessor *Scaffold Tree Generator* (STG)—and they can only generate scaffold trees. SNG improves on SH and STG in several critical ways.

(1) SNG can generate both scaffold networks and trees.

(2) SNG is a command-line tool that may be incorporated into an automated drug discovery pipeline, while SH requires manual user interface.

(3) SNG can generate trees or networks on significantly larger datasets than SH. SNG runs to completion on a dataset of 10 000 000 molecules, while SH fails to complete on a dataset of 225 000 molecules (Tests were performed on a Mac Intel Core i5 2.7 GHz with 8 Gb of RAM. JVM was set to a maximum memory allowance of 2 Gb using -Xmx2048m option for all tested applications.).

(4) SNG is significantly faster than SH—for a 150 000 molecule dataset SNG takes 37 m 3 s and SH takes 50 m to generate a scaffold tree (Tests were performed on a Mac Intel Core i5 2.7 GHz with

*To whom correspondence should be addressed.

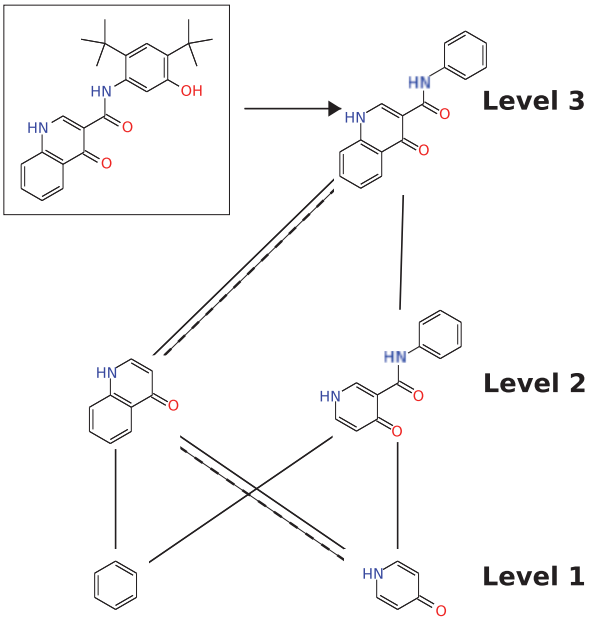

Fig. 1. The scaffold network for Invacaftor—the molecule boxed in the upper left—shows all possible ways the drug can be decomposed into composite scaffolds. The second molecule in Level 3 is the Bemis–Murcko scaffold of Invacaftor, while the sub-scaffolds at Levels 1 and 2 represent the removal of a peripheral ring and resulting terminal side-chains from their connected parent scaffold. The double-dotted lines show the specific way that the *Scaffold Tree* algorithm decomposes Invacaftor. SNG can compute either the full network or just the tree. Molecules with the same scaffold, or scaffolds connected in the network or tree, tend to have the same biological activity

8 Gb of RAM. JVM was set to a maximum memory allowance of 2 Gb using -Xmx2048m option for all tested applications.).

(5) SNG can be run in parallel on multiple input files and can aggregate the networks or trees from multiple runs into a consensus network, dramatically improving runtime. When SNG is run using 4 parallel processes on the same 150 000 molecule dataset mentioned above the runtime is 17 m 13 s (Tests were performed on a Mac Intel Core i5 2.7 GHz with 8 Gb of RAM. JVM was set to a maximum memory allowance of 2 Gb using -Xmx2048m option for all tested applications.).

(6) SNG solves numerous bugs present in SH—including hanging indefinitely on certain molecules that contain large numbers of rings and occasionally replacing double bonds in heteroaromatic rings with single bonds when analysing molecules in SMILES format—and STG—a canonization error whereby two structurally identical scaffolds have different SMILES representations, and therefore multiple nodes in the same tree.

2.1 Implementation

SNG is written in Java, allowing it to be run on virtually any computing platform. It incorporates code from the SH software package and the Chemistry Development Kit (Steinbeck *et al.*, 2003) to enumerate available rings and remove the least characteristic ring, and generate SVG image output of molecular structures. The bond and canonization errors mentioned above stem from failures of the CDK to correctly interpret certain SMILES strings; SNG avoids these errors by employing OpenBabel (O'Boyle *et al.*, 2011) to convert SMILES format input to SDF format before calling the CDK code to generate a scaffold network or tree.

2.2 Input and output

Input structures to SNG may be submitted in either SMILES or SDF format.

SNG produces a scaffold tree or network file in either SDF or tab-delimited text format, which, respectively, describe the structure of each submitted scaffold and its composite sub-scaffolds in either SDF or SMILES format. Each scaffold is described by a unique ID number, the number of rings that the scaffold contains, and the unique IDs of its sub-scaffolds. SNG output for the network in Figure 1 is given in the Supporting Information.

2.3 Performance

SNG is built with substantial fault tolerance and can recover when any individual input molecule causes a failure or is too complex to be processed in a user-specified amount of time (default 30 s). We have evaluated SNG throughput when applied to a large subset of molecules from PubChem. SNG was run on 9 873 306 input molecules from PubChem. This results in a network with 4 897 085 unique scaffolds. Total processing time required for this task was 5 d, 15 h, 51 m. When individual SNG runs were executed in parallel with ~600 jobs and the resulting networks were subsequently aggregated to create the consensus network of 4 897 085 unique scaffolds the total time on a 12 core machine was ~13 h.

3 CONCLUSION

SNG is a new command-line tool for use in drug discovery. SNG extracts scaffold networks from large sets of molecules. These networks can be incorporated into an automated pipeline for visualizing and analysing large databases of chemical information in order to identify scaffolds that can be developed into drugs.

ACKNOWLEDGEMENTS

SNG incorporates code from *Scaffold Hunter*, the Chemistry Development Kit, and relies upon external calls to OpenBabel. Both M.K.M. and J.M.Z. contributed equally to this article. M.K.M. implemented SNG, revised the manuscript and wrote the online description of the SNG tool; J.M.Z. wrote the initial manuscript; and S.J.S. provided project oversight and substantive editing. We like to thank the Washington University School of Medicine and Department of Pathology and Immunology for resources and support.

Conflict of Interest: none declared.

REFERENCES

Bemis,G.W. and Murcko,M.A. (1996) The properties of known drugs. 1. Molecular frameworks. *J. Med. Chem.*, **39**, 2887–2893.

O'Boyle,N. *et al.* (2011) Open Babel: an open chemical toolbox. *J. Cheminform.*, **3**, 1–14.

Proudfoot,J. (2002) Drugs, leads, and drug-likeness: an analysis of some recently launched drugs. *Bio. Med. Chem. Lett.*, **12**, 1647–1650.

Schuffenhauer,A. *et al.* (2007) The scaffold tree – visualization of the scaffold universe by hierarchical scaffold classification. *J. Chem. Inf. Model.*, **47**, 47–58.

Steinbeck,C. *et al.* (2003) The Chemistry Development Kit (CDK): an open-source java library for chemo- and bioinformatics. *J. Chem. Inf. Comp. Sci.*, **43**, 493–500.

Swamidass,S.J. *et al.* (2012) Utility-aware screening with clique-oriented prioritization. *J. Chem. Inf. Model.*, **52**, 29–37.

Varin,T. *et al.* (2011) Mining for bioactive scaffolds with scaffold networks: improved compound set enrichment from primary screening data. *J. Chem. Inf. Model.*, **51**, 1528–1538.

Wetzel,S. *et al.* (2009) Interactive exploration of chemical space with Scaffold Hunter. *Nat. Chem. Biol.*, **1875**, 581–583.