# `cblaster`: a remote search tool for rapid identification and visualisation of homologous gene clusters

Cameron L.M. Gilchrist*[1], Thomas J. Booth*[1], Yit-Heng Chooi*[1]

[1]School of Molecular Sciences, The University of Western Australia, 35 Stirling Hwy, Crawley, 6009

E-mail: cameron.gilchrist@research.uwa.edu.au, thomas.booth@uwa.edu.au, yitheng.chooi@uwa.edu.au

## Abstract

Genes involved in coordinated biological pathways, including metabolism, drug resistance and virulence, are often collocalised as gene clusters. Identifying homologous gene clusters aids in the study of their function and evolution, however existing tools are limited to searching local sequence databases. Tools for remotely searching public databases are necessary to keep pace with the rapid growth of online genomic data. Here, we present `cblaster`, a Python based tool to rapidly detect collocated genes in local and remote databases. `cblaster` is easy to use, offering both a command line and a user-friendly graphical user interface (GUI). It generates outputs that enable intuitive visualisations of large datasets, and can be readily incorporated into larger bioinformatic pipelines. `cblaster` is a significant update to the comparative genomics toolbox. `cblaster` source code and documentation is freely available from GitHub under the MIT license (github.com/gamcil/cblaster).

# Introduction

Complex biological processes are coordinated through the action of multiple genes, often found collocated on the genome as gene clusters. Gene clusters have been extensively studied in microbes for their ability to encode traits such as the biosynthesis of secondary metabolites, virulence, drug resistance and xenobiotic degradation; however, gene clustering is a phenomenon observed across all kingdoms of life.[1;2;3] Given the significance of these processes in medicine and agriculture, it is important to understand the genetic mechanisms that underpin them.

Tools such as MultiGeneBlast[4] or clusterTools,[5] take specific queries (e.g. nucleotide/protein sequences, Hidden Markov Model (HMM) profiles) and search them against genomic datasets, looking for any instances of collocation. However, their utility is limited by the requirement to build and maintain local databases. In many cases, this is redundant given that most biological sequence data is deposited in publicly accessible online databases like the National Center for Bioinformatics Information (NCBI). It is also increasingly demanding as the amount of available data has grown rapidly in the past decade. Between January 2010 and September 2020, the total number of RefSeq genomes grew from 10,171 to 104,969 (NCBI RefSeq Growth Statistics page). This has two additional consequences: i) maintaining an up-to-date local database has become significantly more onerous; and ii) the size of search outputs has increased significantly. Therefore there is a need for search tools that can leverage remote databases and visualise search outputs in an intuitive manner.

Here, we present `cblaster`, a Python-based tool to rapidly search for collocated protein coding regions remotely by leveraging NCBI APIs, or locally within user-generated databases. `cblaster` is easy to use with both command line and graphical user interfaces (GUI). It generates fully interactive visualisations implemented in JavaScript and HTML that allow the user to intuit patterns from complex datasets, as well as textual outputs that can easily be incorporated into bioinformatic pipelines. `cblaster` is a significant update to the comparative genomic toolbox and is a launch pad for further functional and evolutionary analyses.

# Fully remote searches against NCBI sequence databases

The `cblaster` search workflow is detailed in Figure 1 and can be launched either through the command line interface (Table 1) or the GUI (Figure 2). A search begins with the user providing either a FASTA file containing protein sequences, or a collection of valid NCBI sequence identifiers (i.e. accessions, GI numbers). If the latter is provided, sequences corresponding to each identifier are downloaded

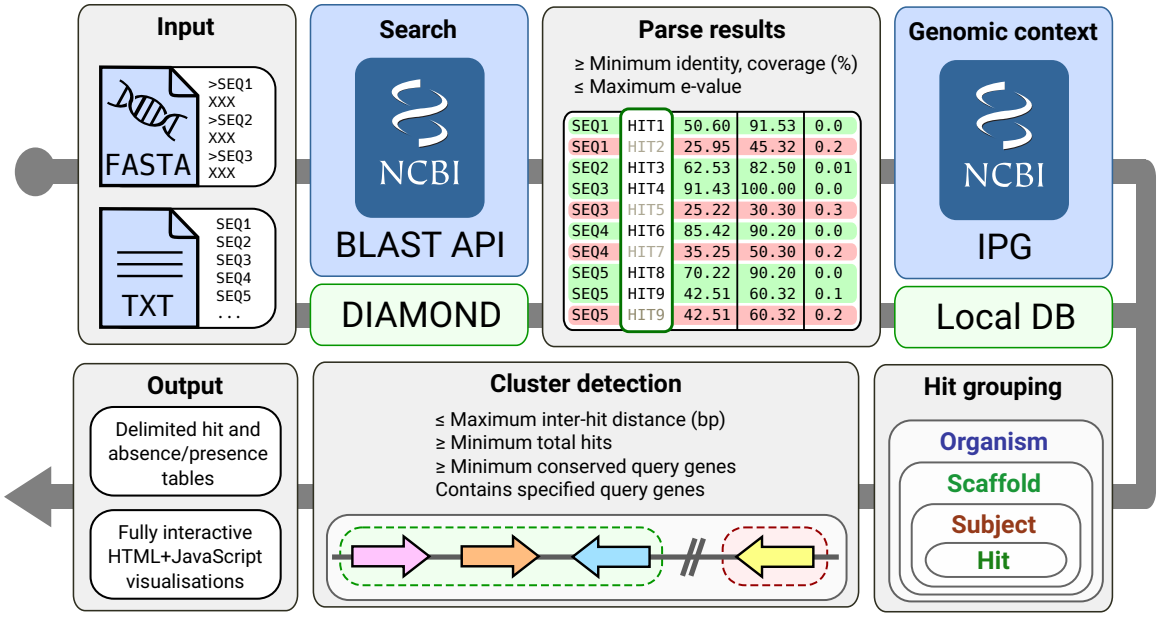


Figure 1: The `cblaster` search workflow. Input sequences are given either as a FASTA file or as a text file containing NCBI sequence accessions. They are then searched against the NCBI's BLAST API or a local DIAMOND database, in remote (blue background) and local (green background) modes, respectively. BLAST hits are filtered according to user defined quality thresholds. Genomic coordinates for each hit are retrieved from the Identical Protein Groups (IPG) resource. Hits are grouped by their corresponding organism, scaffold and subjects. Finally, hit clusters are detected in each scaffold and results are summarised in output tables and visualisations.

Table 1: Overview of modules provided by the `cblaster` command-line interface.

| Module | Description |
|---|---|
| `cblaster search` | The main `cblaster` search workflow |
| `cblaster makedb` | Create local search databases from personal sequence data |
| `cblaster gne` | Gene neighbourhood estimation of `cblaster` search results |
| `cblaster extract` | Extract sequences from `cblaster` search results |
| `cblaster gui` | Launcher for the `cblaster` graphical user interface (GUI) |

using the NCBI Entrez API[6] prior to starting the search. These sequences are uploaded to the NCBI BLAST API to launch a new search using BLASTp from the NCBI BLAST+ suite.[7;6] Optionally, an Entrez search query can be provided to pre-filter the target search database, for example to specify a taxonomic group of interest. The BLAST API assigns a unique request identifier (RID) to the search which remains active for 36 hours. `cblaster` can retrieve remote BLAST search results given a valid RID. `cblaster` can also save search sessions, which can be loaded back into the program for additional filtering, without the need to repeat entire searches.

The BLAST API is repeatedly polled until the search has completed, at which point search results are downloaded and filtered based on user-defined hit quality thresholds (minimum identity, coverage and maximum e-value). To retrieve the genomic context of remote BLAST hits, `cblaster` leverages the Identical Protein Groups (IPG) resource[6] via the Entrez API. The IPG database stores links

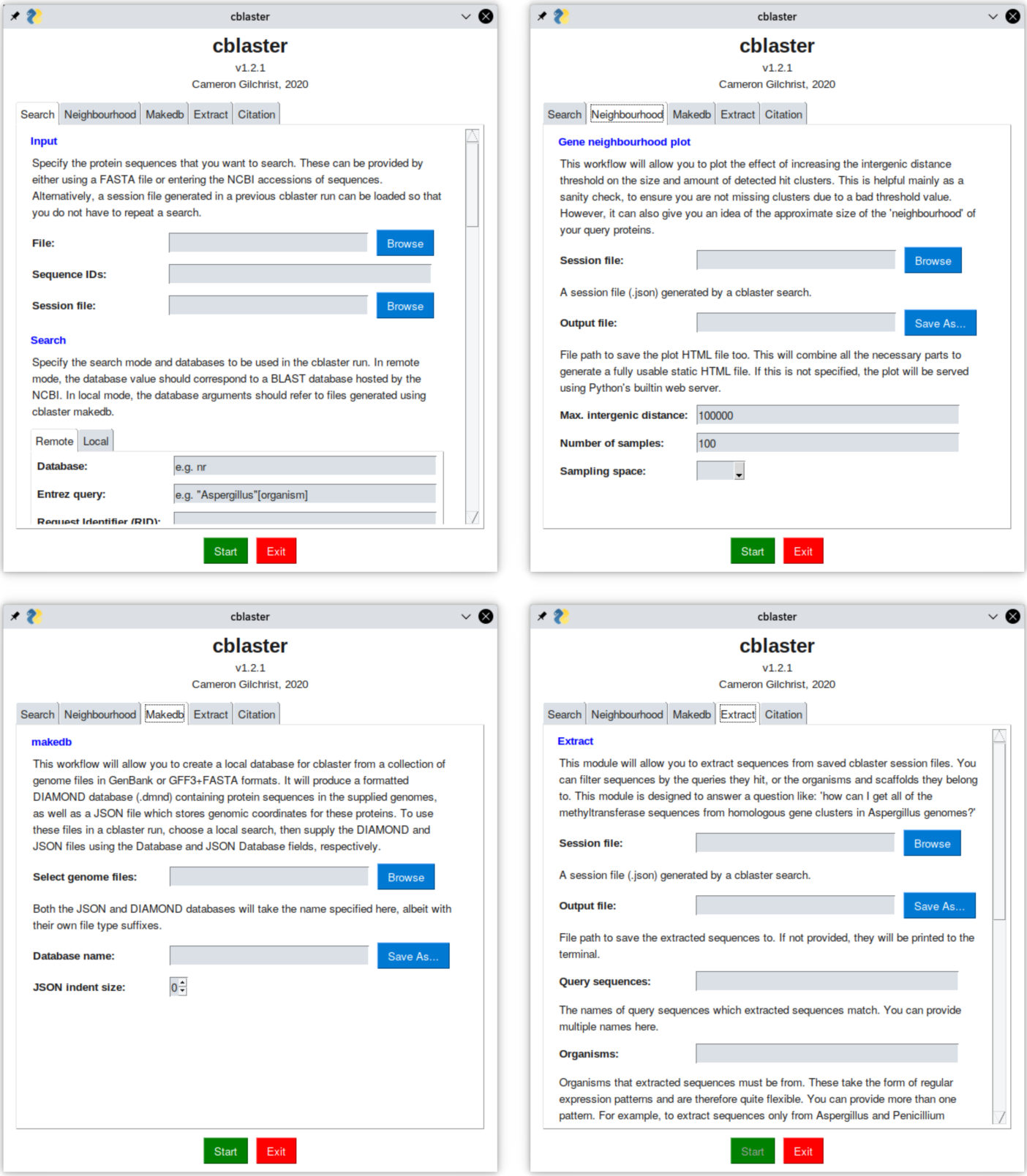


Figure 2: The `cblaster` graphical user interface (GUI). Each panel reprisents a single `cblaster` module: `cblaster search` (top left) for perfoming searches against remote and local databases; `cblaster gne` (top right) for performing genomic neighbourhood estimation; `cblaster makedb` (bottom left) for building local databases from GenBank files and; `cblaster extract` (bottom right) for extracting FASTA files of specific groups of homologues.

from proteins to their exact genomic coordinates, meaning BLAST hits can be traced back to their genomic origins. To ensure no potential clusters are missed, for example due to annotation error or fragmented genome assembly, all rows in the IPG table are saved and linked back to their original hits. Hits are grouped by their subject proteins, which are in turn grouped by genomic scaffold and organism. Finally, subjects are grouped into clusters if they contain hits to any required query sequences specified by the user (if any), and satisfy thresholds for maximum intergenic distance, minimum size and minimum hits per unique query sequence.

# Local searches against custom sequence databases

`cblaster` can also search local search databases. The local search workflow mirrors that of remote searches, with two key differences: i) the DIAMOND search tool[8] is used instead of BLASTp; and ii) a local genome database is used in place of the IPG database to facilitate retrieval of the genomic coordinates of hit proteins.

A local search thus requires two databases to be created by the user: a formatted DIAMOND search database and a local genome database. `cblaster` provides a module, `makedb`, which can rapidly generate both databases from a collection of genome sequences simultaneously in a single command. Briefly, individual GenBank or GFF3 genome files are parsed by the `genome2json` package,[9] which converts them into a hierarchical JSON format. `cblaster` then consolidates the parsed genomes into a single database file. Protein translations from each genome are extracted to a FASTA file from which the DIAMOND database is built. Each sequence is given a unique header consisting of the numeric database indices of each sequence, as well as their corresponding scaffolds and organisms, which makes it possible for `cblaster` to easily retrieve information pertaining to sequences hit during the search.

# Estimation of genomic neighbourhood size

During cluster detection, `cblaster` uses a user-defined threshold to determine the maximum distance between any two BLAST hits in a cluster. By default, a cluster is finalised if no new hit is found within 20Kbp of the previous hit. However, this value is arbitrary and may be inappropriate for some datasets, and so it is advisable to test the effect of changing this parameter on each dataset being analysed. `cblaster` provides the `gne` module, which leverages its ability to rapidly reload and recompute search sessions to robustly automate this analysis. When provided with a session file, `gne`

iteratively performs cluster detection over a user-determined range of intergenic distance threshold values, then plots the total number of predicted clusters, as well as the mean and median cluster size (bp), at each value (Figure 3b; Additional File 2). These plots typically resemble logarithmic growth, steeply rising at low values but gradually levelling off at higher values. This makes it easy to determine sensible cutoffs that avoid liminal regions where small variability would have a large effect on the output.

# Comprehensive output and fully interactive visualisations

`cblaster` offers rich summaries and visualisations of search results. `cblaster`'s outputs are available in human-readable and character-delimited formats, and can be easily incorporated into higher level bioinformatic pipelines. This includes a summary of all detected clusters and a presence/absence table (here termed binary table), which shows the total number of hits in detected clusters per query sequence. Also, the `extract` module can be used to retrieve specific homologue sequences from search results for further analyses.

Search results are visualised as a cluster heatmap (Figure 3a; Additional File 1). Clusters are hierarchically clustered based on best hit identity values using the SciPy library.[10] Cells in the heatmap are shaded based on identity. The text inside each cell indicates if query sequences have multiple hits within a cluster. Figures can be freely panned and zoomed; columns (query sequences) and rows (clusters) can be hidden by clicking on their respective labels. Additionally, mousing over a cell in the cluster heatmap generates a tooltip that displays a summary of hits in the cluster for the corresponding query sequence. The tooltip also links to the exact cluster location on the NCBI's graphic genome viewer.

Similarly, `cblaster gne` results are visualised as line charts (Figure 3b) which can be panned and zoomed, with a tooltip showing the number of detected clusters, as well as the mean and median cluster size (bp), at different intergenic distance values.

`cblaster` visualisations are implemented as HTML documents that can be opened in the web browser. These documents contain scalar vector graphics (SVG) images generated using the D3 visualisation library,[11] which can be exported for modification in vector image manipulation software. Any changes made within the visualisation are reflected in the exported image. Fully portable HTML documents can be generated, enabling results to be shared between different computers.

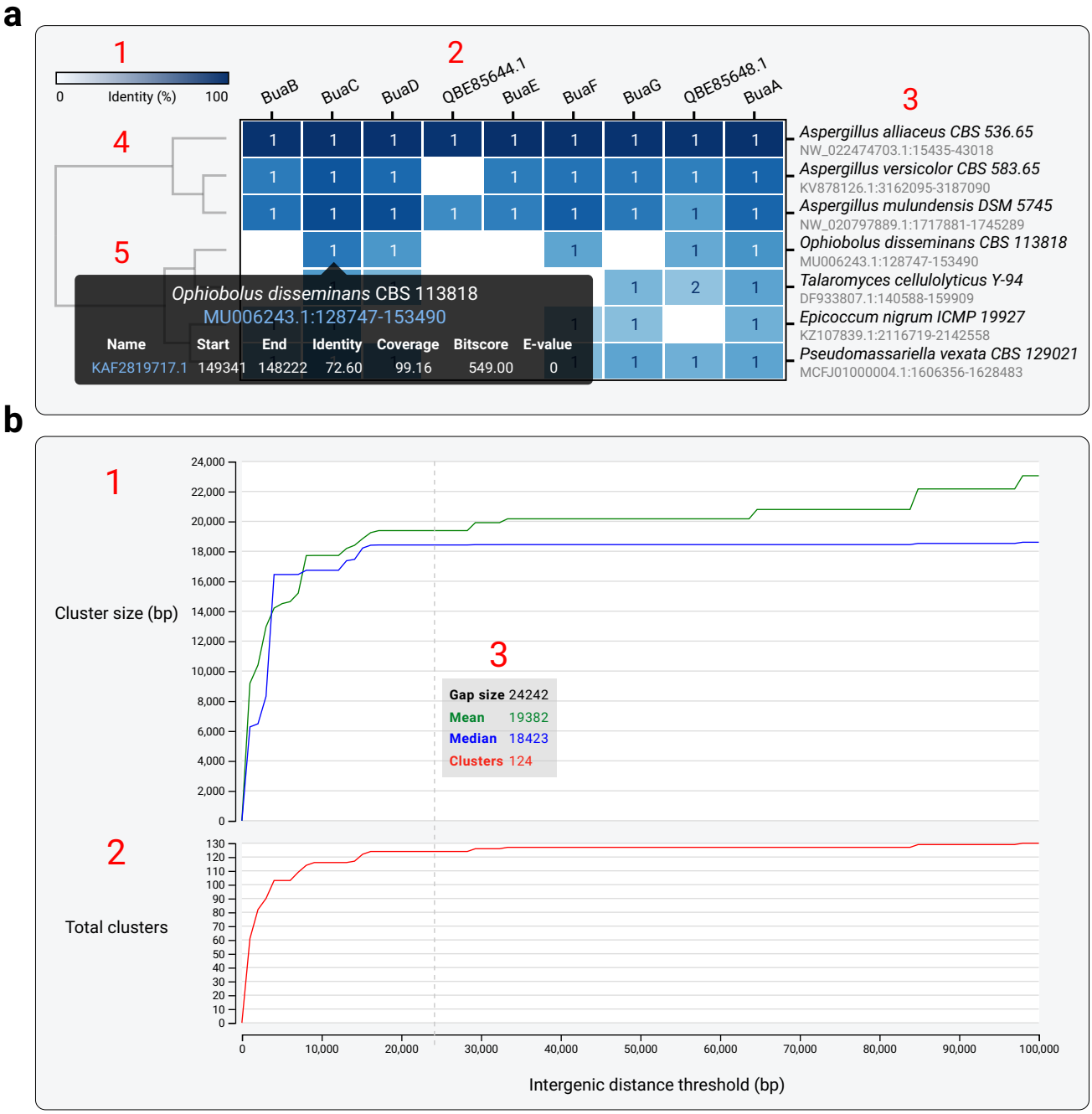


Figure 3: Interactive visualisations generated by `cblaster`. a) Cluster heatmap visualisation of `cblaster` search results: 1) heatmap colour bar indicating 0% (white) to 100% (blue) identity; 2) names of query sequences; 3) names of organism and scaffold locations of hit clusters; 4) dendrogram of hit clusters generated from their identity to query sequences; and 5) cell hover tooltip with detailed hit information including hyperlinks to genomic position on NCBI. b) Gene neighbourhood estimation (GNE) visualisation: 1) Plot of mean and median hit cluster sizes (bp) at different gap sizes; 2) Plot of total clusters at different gap sizes; 3) Hover tooltip showing values of mean and median cluster size (bp) and total clusters at a given gap size.

Table 2: Feature comparison of software tools for gene cluster homology searches.

| **Feature** | **cblaster** | **MultiGeneBlast** | **clusterTools** |
|---|---|---|---|
| Query input | Proteins | Proteins, nucleotides | HMM profiles |
| Searches against remote databases | X | | |
| Genomic neighbourhood estimation | X | | |
| Extraction of hit sequences | X | | |
| Interactive visualisations | X | X | |
| Allows input of personal sequence data | X | X | X |
| Allows creation of local search databases | X | X | X |
| Graphical user interface available | X | X | X |

# New approaches and practical applications

`cblaster` is a powerful tool for identifying homologous gene clusters from NCBI or local databases. It is possible to query known gene clusters to expand existing families, or user-specified combinations of genes that are predicted to function together. Currently, the most widely used tools for conducting gene cluster homology searches, MultiGeneBlast[4] and clusterTools,[5] depend on local sequence databases. `cblaster` allows remote searching of NCBI sequence databases, eliminating this requirement and reducing the computational burden (Table 2).

`cblaster` provides outputs that can be easily adapted for downstream analyses. Delimited formats can be accessed as spreadsheets, making them accessible to all researchers regardless of bioinformatics experience. The binary table can be sorted and filtered post-analysis, which is particularly useful for narrowing large datasets to focus on individual genes or organisms of interest. In this way, data can be easily extracted to generate simple graphs (e.g. generic distribution, ubiquity of different homologues or average cluster size), build distance matrices and dendrograms. Results tables from multiple searches can easily be concatenated to combine results from local and remote searches. `cblaster extract` can be used to quickly obtain specific homologue sequences for downstream phylogenetic and structural analyses.

By using heatmaps, as opposed to more complex gene diagrams, information about the diversity and conservation of encoded proteins is accessible at a glance (Figure 3a). Grouping gene clusters by identity highlights blocks of homologous proteins, facilitating the designation of cluster boundaries and identification of core biosynthetic genes and functional sub-clusters. This is a useful proxy for functional and evolutionary relationships. For example, we recently used this technique to group biosynthetic gene clusters of two closely related natural products, the indolotryptolines and indolocarbazoles (Supplementary Information Case Study 1; Additional File 1). This was possible knowing only the presence or absence of genes within a given cluster. In this manner, `cblaster` can uncover

deep-seated evolutionary relationships that may not be immediately apparent, such as proteins or sub-clusters that have been co-opted between different pathways.

`cblaster gne` can estimate the size of genomic neighbourhoods to help evaluate the robustness of cluster prediction (Figure 3b). This enables the user to identify suitable intergenic distance values that limit missing data without inadvertently introducing false positives. In addition, recent work has highlighted the importance of genomic neighbourhoods in the evolution of gene clusters.[12] The distribution of genomic neighbourhood sizes, as calculated by `gne`, provides insight into the conservation of gene cluster topology; a broader distribution indicates a more dynamic architecture. This aspect is likely to be of particular interest in the study of plants and other eukaryotes where gene clusters are more variable in gene density. For example, to test the utility of the module, we analysed the neighbourhood sizes of triterpene biosynthetic gene clusters in plants (Supplementary Information Case Study 2; Additional File 2).

`cblaster` can uncover homologous gene clusters and evolutionary relationships at a pace that far exceeds other available software and has become an indispensable tool in our own research into the biosynthetic gene clusters of fungi and bacteria. It has already aided us in the identification of several metabolic gene clusters from *Aspergillus* spp. (Supplementary Information Case Study 3; Additional File 3).[13;14;15] We foresee `cblaster` being used to uncover novel variants of known pathways, or to prospect for new combinations of biological parts for use in synthetic biology.[16;17;18] `cblaster`'s versatility means it can be applied to any project that requires the study of two or more collocated genes and it is accessible to all researchers regardless of their experience with bioinformatics.

# Software implementation and availability

`cblaster` is implemented in Python, with visualisations written in JavaScript and HTML. Source code, documentation and usage information is freely distributed on GitHub (github.com/gamcil/cblaster) under the MIT license.

# Acknowledgements

C.L.M.G. is supported by the Australian Government Research Training Program Ph.D. scholarship. Y-H.C is supported by an Australian Research Council Future Fellowship (FT160100233). This work was funded in part by the Cooperative Research Centres Projects scheme (CRCPFIVE000119). We

thank Dr. Marnix H. Medema for his useful feedback on the manuscript and program.

# References

[1] Marc G. Chevrette, Karina Gutiérrez-García, Nelly Selem-Mojica, César Aguilar-Martínez, Alan Yañez-Olvera, Hilda E. Ramos-Aboites, Paul A. Hoskisson, and Francisco Barona-Gómez. Evolutionary dynamics of natural product biosynthesis in bacteria. *Nat Prod Rep*, 37(4):566–599, April 2020. ISSN 1460-4752. doi: 10.1039/C9NP00048H.

[2] Shengqin Wang, Zhihong Zheng, Huixi Zou, Nan Li, and Mingjiang Wu. Characterization of the secondary metabolite biosynthetic gene clusters in archaea. *Comput Biol and Chem*, 78:165–169, February 2019. ISSN 1476-9271. doi: 10.1016/j.compbiolchem.2018.11.019.

[3] Hans-Wilhelm Nützmann, Claudio Scazzocchio, and Anne Osbourn. Metabolic Gene Clusters in Eukaryotes. *Annu Rev of Genet*, 52(1):159–183, November 2018. ISSN 0066-4197, 1545-2948. doi: 10.1146/annurev-genet-120417-031237.

[4] Marnix H. Medema, Eriko Takano, and Rainer Breitling. Detecting Sequence Homology at the Gene Cluster Level with MultiGeneBlast. *Mol Biol Evol*, 30(5):1218–1223, May 2013. ISSN 1537-1719, 0737-4038. doi: 10.1093/molbev/mst025.

[5] Emmanuel Lorenzo de los Santos and Gregory Challis. clusterTools: Functional element identification for the in silico prioritization of biosynthetic gene clusters. *Access Microbiol*, 1(1A), 2019. doi: 10.1099/acmi.ac2019.po0154.

[6] NCBI Resource Coordinators. Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 45(D1):D12–D17, January 2017. ISSN 0305-1048. doi: 10.1093/nar/gkw1071.

[7] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L. Madden. BLAST+: Architecture and applications. *BMC Bioinformatics*, 10:1–9, 2009. ISSN 14712105. doi: 10.1186/1471-2105-10-421.

[8] Benjamin Buchfink, Chao Xie, and Daniel H Huson. Fast and sensitive protein alignment using DIAMOND. *Nat Methods*, 12(1):59–60, jan 2015. ISSN 1548-7091. doi: 10.1038/nmeth.3176. URL `http://www.nature.com/articles/nmeth.3176`.

[9] Cameron L. M. Gilchrist. genome2json v0.1.1. *Zenodo*, 2020. doi: 10.5281/zenodo.4063351.

[10] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, Yoshiki Vázquez-Baeza, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2.

[11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 Data-Driven Documents. *IEEE T Vis Comput Gr*, 17(12):2301–2309, December 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2011.185.

[12] Zhenhua Liu, Hernando G. Suarez Duran, Yosapol Harnvanichvech, Michael J. Stephenson, M. Eric Schranz, David Nelson, Marnix H. Medema, and Anne Osbourn. Drivers of metabolic diversification: How dynamic genomic neighbourhoods generate new biosynthetic pathways in the Brassicaceae. *New Phytol*, 227(4):1109–1123, 2020. ISSN 1469-8137. doi: 10.1111/nph.16338.

[13] Hang Li, Cameron L.M. Gilchrist, Heather J. Lacey, Andrew Crombie, Daniel Vuong, John I. Pitt,

Ernest Lacey, Yit Heng Chooi, and Andrew M. Piggott. Discovery and Heterologous Biosynthesis of the Burnettramic Acids: Rare PKS-NRPS-Derived Bolaamphiphilic Pyrrolizidinediones from an Australian Fungus, Aspergillus burnettii. *Org Lett*, 21(5):1287–1291, 2019. ISSN 15237052. doi: 10.1021/acs.orglett.8b04042.

[14] Heather J. Lacey, Cameron L. M. Gilchrist, Andrew Crombie, John A. Kalaitzis, Daniel Vuong, Peter J. Rutledge, Peter Turner, John I. Pitt, Ernest Lacey, Yit-Heng Chooi, and Andrew M. Piggott. Nanangenines: Drimane sesquiterpenoids as the dominant metabolite cohort of a novel Australian fungus, Aspergillus nanangensis. *Beilstein J Org Chem*, 15(1):2631–2643, November 2019. ISSN 1860-5397. doi: 10.3762/bjoc.15.256.

[15] Hang Li, Cameron L.M. Gilchrist, Chin Soon Phan, Heather J. Lacey, Daniel Vuong, Stephen A. Moggach, Ernest Lacey, Andrew M. Piggott, and Yit Heng Chooi. Biosynthesis of a New Benzazepine Alkaloid Nanangelenin A from Aspergillus nanangensis Involves an Unusual l-Kynurenine-Incorporating NRPS Catalyzing Regioselective Lactamization. *J Am Chem Soc*, 142(15):7145–7152, apr 2020. ISSN 15205126. doi: 10.1021/jacs.0c01605. URL `https://dx.doi.org/10.1021/jacs.0c01605`.

[16] Yen Hsiang Wang, Kathy Y. Wei, and Christina D. Smolke. Synthetic biology: Advancing the design of diverse genetic systems. *Annu Rev Chem Biomol*, 4:69–102, 2013. ISSN 19475438. doi: 10.1146/annurev-chembioeng-061312-103351. URL `/pmc/articles/PMC3773533/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC3773533/`.

[17] Huihua Sun, Zihe Liu, Huimin Zhao, and Ee Lui Ang. Recent advances in combinatorial biosynthesis for drug discovery, feb 2015. ISSN 11778881. URL `/pmc/articles/PMC4334309/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4334309/`.

[18] Cameron L. M. Gilchrist, Hang Li, and Yit-Heng Chooi. Panning for gold in mould: Can we increase the odds for fungal genome mining? *Org Biomol Chem*, 16(10):1620–1626, 2018. ISSN 1477-0520, 1477-0539. doi: 10.1039/C7OB03127K.