Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Genome analysis

OXFORD

Supplementary Material

Genome Context Viewer: visual exploration of multiple annotated genomes using microsynteny

Alan Cleary [1, 2, †,*] and Andrew Farmer [2, †]

[1] Gianforte School of Computing, Montana State University, Bozeman, MT, 59717, United States and
[2] National Center for Genome Resources, Santa Fe, NM, 87505, United States.

[*] To whom correspondence should be addressed.
[†] The authors wish it to be known that they should be regarded as joint First Authors.

Abstract

Summary: The Genome Context Viewer (GCV) allows users to search, align, and visualize multiple regions of genomes considered primarily with respect to the ordering and orientation of their annotated gene content. Here we discuss the GCV architecture, technology stack, and the services that it consumes, as well as some of the algorithms used in service implementations and in the client.
Availability and implementation: GCV is provided under the GNU General Public License version 3 (GPL-3.0). Source code is available at https://github.com/legumeinfo/lis_context_viewer.
Contact: alan.cleary@msu.montana.edu

GCV is currently in use at the websites of the Legume Information System (LIS, http://legumeinfo.org/lis_context_viewer) (Dash *et al.*, 2016) and the Legume Federation (LegFed, http://legumefederation.org/lis_context_viewer). Non-plant examples can be found at https://github.com/legumeinfo/lis_context_viewer/wiki/Examples. In the following section we describe how GCV is integrated into these sites, and in the sections thereafter we describe the GCV architecture and algorithms, using LIS and LegFed as examples. We conclude with a discussion of related work.

1 LIS and LegFed Integration

The Legume Information System is an online platform for legume breeders and researchers that houses a variety of genetic and genomic data of model legume species relevant to industrial agriculture. The Legume Federation is a consortium of legume researchers and groups, including LIS, with the objective of fostering the adoption of data standards, distributed development, and enabling comparative analyses. GCV was born from the needs of these projects and is integrated into the sites as follows.

LIS acquires annotated genomes from a variety of independently managed projects and primary data repositories. In accordance with the mission of the Legume Federation, homology relationships among the genes annotated in these genomes are established by initial HMM-based assignment to the Phytozome Angiosperm-level gene families (Goodstein *et al.*, 2012); this in itself is sufficient for the purposes of making genomes available for use in GCV. Further steps of multiple sequence alignment and phylogenetic gene tree construction are used to produce interactive tree visualizations which interoperate with GCV in several ways. For example, a collection of genes can be specified from an arbitrary subtree and loaded into the basic view of GCV as the set of focus genes from which context tracks are derived. Alternatively, a leaf node of a phylogeny can be used as the focus gene of a query track in the search view of GCV. This linking can also be reversed, that is, in GCV the user may select a gene family and view the phylogenetic tree for that family with the members present in the linking GCV context highlighted.

Given the vast amount and different types of data housed by LIS, a heterogeneous collection of software is used to facilitate user exploration and knowledge discovery. As illustrated by the interoperability of the phylogenies and GCV, interlinking between these software is crucial to the utility of LIS. As such, GCV is configured to link to various LIS tools including Tripal gene pages (Cho *et al.*, 2012) and InterMine reports (Smith *et al.*, 2012). Additionally, a service is used to link to external tools specific to the sources from which the various genomes were acquired. This external linking service is further described in Section 2.2.

2 Architecture

GCV is a client-side single page Web application and so can be run locally or served as part of a website. It consumes data from one or more providers that implement the interface defined in a RESTful API; see https://github.com/legumeinfo/lis_context_viewer for the full API. Once it has aggregated data from the providers, GCV creates visualization-specific data representations (micro-synteny alignments, dot plots, and macro-synteny tracks), filters these data according to user-defined criteria, and

visualizes the results. The software architecture and data flow is depicted in Figure 1.

2.1 Technology Stack

GCV was implemented using modern web standards technologies, specifically, Angular 2 (https://angular.io/), ngrx/store (https://github.com/ngrx/store), and D3 (https://d3js.org/). Angular 2 is used to retrieve data from service providers, manage UI components, and mediate communication between the visualizations and the rest of the application. ngrx/store is used to manage application state and give GCV explicit data flow, and D3 is used to draw the various visualizations. Furthermore, the visualizations and their inter-visualization interaction mechanisms have been encapsulated in their own JavaScript library and so can be used independently of GCV.

GCV is service implementation agnostic, only requiring that the services it uses adhere to the RESTful API. Even so, the service implementation used by LIS and LegFed is freely available so that users with similar infrastructures need not re-implement GCV services. It is implemented as a Django application (https://www.djangoproject.com/) on top of a Chado database (Mungall *et al.*, 2007).

2.2 Services

The providers that GCV uses must implement one or more of the the RESTful API services. Here we will briefly describe each of the services.

basic micro-synteny tracks: Takes a set of focus genes and the number of neighbors that should flank each gene. It returns the set of tracks centered about the given focus genes.

search micro-synteny tracks: Takes a query track, that is, an ordered list of gene families and search parameters: minimum number of matched families and maximum number of non-matched families between any two matched families. It returns all micro-synteny contexts in the database that meet the given parameter constraints.

gene to query: Takes a focus gene and the number of neighbors as input and passes them to the *basic micro-synteny tracks* service. It then returns the resulting basic track.

macro-synteny tracks: Takes a reference chromosome and a set of aligned chromosomes as input and returns the synteny blocks of the aligned chromosomes with positions relative to the reference chromosome.

nearest gene: Takes a chromosome and a position on the chromosome and returns the gene nearest to that position on the chromosome.

global plot: Takes a set of gene families and a chromosome as input and returns all the genes on the given chromosome that are members of the given gene families.

An important caveat to consider is that the data provided by a service provider may actually be aggregated from multiple curators or data-stores, as is the case with LIS and LegFed. To enable better interoperability with other sites, such as the primary repository of a particular genome, GCV can use services to load links to relevant external sites. For example:

gene links: Takes a gene as input and returns a list of external links that the gene can be passed to for further inquiry.

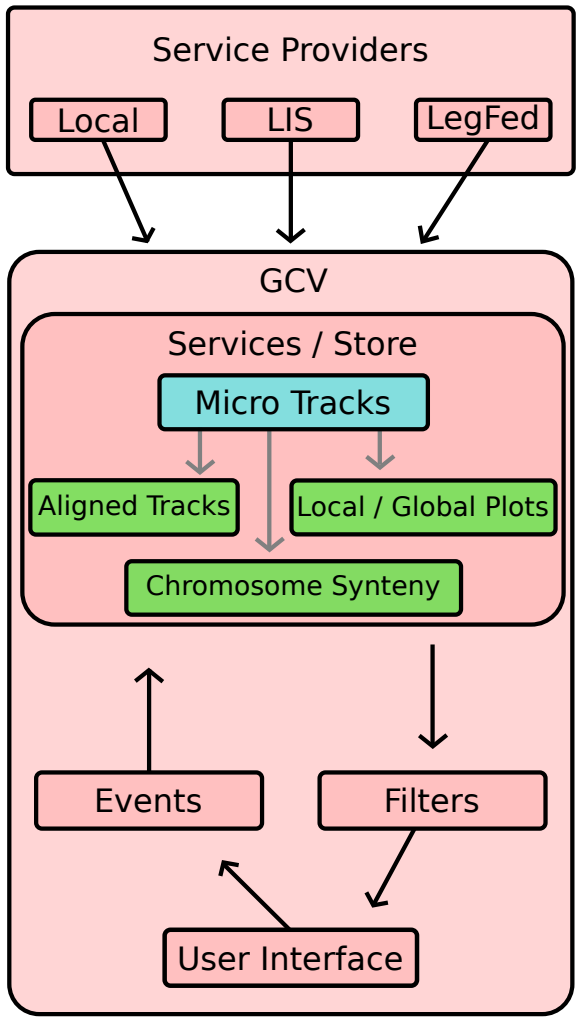

Fig. 1. The software architecture and data flow of GCV. GCV (an Angular 2 application) consumes data from one or more service providers, one of which may be the user's own computer (local). Angular 2 services are responsible for requesting data from service providers, aggregating the results, and storing the results in ngrx/store. In the search view, whenever new micro track data (blue) is acquired, visualization specific representations are made (green). These representations are then filtered by user controlled criteria and consumed by Angular 2 components (the UI) which draw the data with D3. User interaction with the UI can trigger certain events that will notify the services to update the representations or acquire new data, such as changing alignment parameters or search parameters, respectively.

3 Algorithms

A variety of existing and novel algorithms are used both in GCV and the LIS/LegFed services implementation. Here we present four such algorithms that are integral to the Search view and illustrate the data flow depicted in Figure 1.

3.1 Track Search

Although the implementation of GCV services is left to the service providers, due to the search view's central role in GCV and the non-trivial nature of the corresponding service, we will discuss how the open-source example server used by LIS and LegFed implements the micro-synteny search service.

The problem of finding micro-synteny tracks with similar gene family content and ordering to the query track is an instance of the

Fixed-Radius Near Neighbors problem (Bentley, 1975). In the example implementation, the radius is defined by the query parameters - minimum number of *matched* gene families and maximum number of non-matched, or *intermediate*, families between any two matched families - and so the search space is 2-dimensional.

The example implementation uses a heuristic algorithm that works as follows: Given a micro-synteny search query, that is, the list of gene families present in the query track, the algorithm iterates the ordered list of gene families of each chromosome in the database. When a gene family that matches one of the query families is found a new candidate track is created. The candidate is grown by iteratively adding the next family in the list to the track until the number of families added since the last match is greater than *intermediate*. The candidate track is then trimmed back to the last matched family and returned if the number of matched families it contains is greater than or equal to *matched*. The algorithm then continues iterating the ordered list of gene families at the family after the last family iterated by the previous candidate growth.

Since each chromosome in the database may have several thousand genes, the example implementation is optimized as follows: First, the algorithm leverages the indexing mechanisms of the underlying database to efficiently find all instances of the query gene families in the ordered list of gene families for each chromosome and memoizes each matched family's position in its corresponding list. Then the iterative algorithm is applied to each chromosome's ordered list of matched gene families if the list contains at least *matched* number of families. Candidate tracks are grown as before, but the memoized position data is used to compute how many non-matched families lie between the last and next matched families, rather than iterating the non-matched families. A batch query is then performed to fetch all the non-matched families for the candidates that satisfy the *matched* parameter.

Though gene family ordering is important to consider when mining tracks that are syntenic to the query, we leave the consideration of ordering beyond computing the number of genes between matches as a task for the front-end. This is because track similarity in this sense is dictated by the choice of alignment algorithm and corresponding parameter values. Thus the tracks returned by the micro-synteny search service as we have defined it are invariant with respect to choices of alignment algorithm and parameters, and so changing these in the client does not require a new set of server requests.

3.2 Merging Alignments

The Repeat algorithm (Durbin *et al.*, 1998) is an extension of the Smith-Waterman local alignment algorithm (Smith and Waterman, 1981). Specifically, rather than finding the highest scoring local alignment, it finds all local alignments whose score is above a certain *threshold*. In this work we extended the Repeat algorithm to identify inversions whose reversal in the containing sequence improves the sequence's alignment score.

In essence, the extension works by first aligning both the forward and reverse orientations of a sequence to the reference with the Repeat algorithm. All resulting forward alignments are then compared with all reverse alignments for shared gene content. When a reverse alignment is found to have shared gene content with a forward alignment, memoized suffix scores from the alignments' traceback matrices are used to determine if replacing the inverted sequence in the forward alignment with the reverse alignment will improve the forward alignment's score, or vice versa. If so, the forward and reverse alignments are *merged* by replacing the inverted sequence and updating the memoized suffix scores and alignment score.

3.3 Alignment Coordinates

An extension that we have applied to both the Repeat and Smith-Waterman alignment algorithms is the assignment of coordinates to the gene sequences based on their alignments. It works by *positioning* all resulting alignments relative to the reference sequence. Specifically, a matched character is given the position of the character it matched in the reference sequence and inserted characters are given a position between that of the reference characters they were inserted between. For example, given gene family reference $\tau\alpha\kappa\gamma\gamma$ and sequence $\alpha\gamma\kappa\kappa\kappa\gamma$ [1], the following hypothetical forward alignment would be positioned as:

position in reference:		0	1	2	3				4
alignment	reference:	τ	α	κ	γ	-	-	-	γ
	sequence:		α	-	γ	κ	κ	κ	γ
position in reference:			1		3	3.25	3.5	3.75	4

The sequence's forward alignment positioning indicates that it spans the character interval 1-4 in the reference.

By positioning all alignments relative to the reference sequence they are normalized to the same coordinate space. These normalized coordinates are what is used to position the tracks in the micro-synteny search visualization.

3.4 Track Packing

GCV uses the extended Repeat algorithm to detect inversions in micro-synteny search results so that they may explicitly be drawn. In cases where more than one inversion has been found in a single alignment all inversions are to be drawn as compactly as possible, that is, we want to pack as many inversions into as few visualization rows as possible without overlapping any of the inversions. Similarly, in the macro-synteny visualization we want to draw the synteny blocks from the same chromsome as compactly as possible.

This "Track Packing" problem is equivalent to the Interval Partitioning/Coloring problem for which there exists an optimal polynomial-time solution (Kleinberg and Tardos, 2006). The solution works by iteratively applying the greedy polynomial-time solution for the Interval Scheduling problem (Kleinberg and Tardos, 2006) to a set of intervals until all the intervals have been scheduled. In the case of the micro-synteny alignments, the intervals are the position intervals described in Section 3.3. In the case of the macro-synteny blocks, the intervals are the blocks' genomic positions on the reference chromosome.

4 Related Work

The motivation for using gene-families as a unit of search and alignment was two-fold: 1) to emphasize functional content and the genomic contexts in which it occurs and 2) to make these analyses sufficiently performant to be done on-demand across large taxonomic groups in a federated manner. The merit of the first has been sufficiently discussed elsewhere (Lopez and Samuelsson, 2011; Louis *et al.*, 2015) and so we will focus on the second.

The "basic" display of GCV, which is not concerned with similarity of gene content between tracks or with producing track alignments is similar to visualizations provided by other sites such as those seen in the Eukaryotic Gene Order Browser (Lopez and Samuelsson, 2011) or the gene family pages of Phytozome (Goodstein *et al.*, 2012).

It is the GCV's "search" view that makes use of algorithms to determine segmental similarity and collinearity by use of the pre-estabished gene family assignments, and which makes it in some sense comparable to tools that are concerned with problems of whole genome synteny comparison. We note however that the alignments that GCV computes dynamically are not whole genome comparisons, but are limited to the determination of segments within whole genomes that are syntenic to the given query track,

[1] Greek characters are used only for example purposes. In practice the gene families are matched on their unique identifiers.

whose extent determines the computational cost of the subsequent search and alignment problem.

There exist a variety of web-based tools for pairwise and multiple genome synteny search and comparison. The three that are perhaps most related to the GCV are the Plant Genome Duplication Database (PGDD) (Lee *et al.*, 2017), CoGe's GEvo (Lyons *et al.*, 2008), and Genomicus' PhyloView (Louis *et al.*, 2015).

PGDD is a database characterizing whole genome duplication events in plant genomes, providing both intra- and inter-genome syntenic relationships. Similar to the GCV, users can perform synteny searches by providing a query gene or can generate a dot plot by selecting two genomes to compare. A locus search (http://chibba.agtec.uga.edu/duplication/index/locus) will yield a set of precomputed matches between anchor genes for a genomic window of specified size centered on the query gene, rather than an explicitly aligned representation of the query to the resulting tracks (Figure 3).

PGDD uses the MCScanX algorithm (Wang *et al.*, 2012) to precompute collinear blocks, where candidate anchors are based on either pairwise BLAST of the gene models or clustering of genes into homologous groups. Although, as explained above, our algorithmic approach is not geared to producing whole genome alignments, we nevertheless validated the use of our gene family assignments as a surrogate for direct pairwise comparison. Specifically, we compared the results of MCScanX_h algorithm on our gene family assignment derived homologous groups to the results of basic MCScanX on pairwise BLASTs of the CDS for the same genome. For a self-comparison of the Glycine max genome, the latter puts %66 of all gene models into 1110 blocks ranging from 5 to 1074 genes in extent. Using an unfiltered set of gene families, the comparable procedure yields %73 of genes in 2825 blocks ranging in size from 5 to 1102 genes. Inspection of the blocks produced with this approach suggested that many of the excess blocks were the results of genes from massively expanded families residing in large clusters. Since our algorithms for finding similar regions require that the families matched be distinct, we repeated the comparison by filtering our families to exclude from consideration any that contained more than 20 members from Glycine max. The results of this procedure were more similar to the BLAST-based approach with %61 of genes placed into 1319 blocks ranging in size from 6 to 983 genes. Most of these blocks are coequal in extent to the corresponding blocks from the BLAST-based procedure, but are missing some internal anchor genes due to the pre-filtering procedure employed in this context; in GCV's micro-synteny search implementation, these high copy gene families would still be scored as matches during the alignment procedure used on the resulting blocks. Furthermore, the macro-syntenic blocks produced by MCScanX on the gene family assignments are quite similar to those precomputed by a pipeline based on DAGChainer (Haas *et al.*, 2004) and used in the LIS implementation of the GCV (see Figure 2).

CoGe's GEvo (Genome Evolution Analysis) is one of a large suite of tools for genome synteny analysis available through http://genomeevolution.org. GEvo is the tool within CoGe that is probably most comparable to GCV, as users can perform a search by specifying some genomic feature and flanking region as a search query and selecting what alignment algorithm(s) and corresponding parameters to use. Results are then displayed as pairwise mappings of matched blocks between sequences. The query and search sequences can be loaded from CoGe's database, NCBI, or provided by the user; similarly, CoGe allows any user to upload any number of genomes to use as the target to their analyses, which allows it to be used regardless of the specific taxonomic focus of any given user. A GEvo search can also be initiated from a pair of genes previously determined to reside in a syntenic region by one of the other tools in the CoGe suite, e.g. from a dot in the whole genome comparison dotplots produced by SynMap, which makes it convenient as a mechanism for generating sequence-level similarity in restricted regions

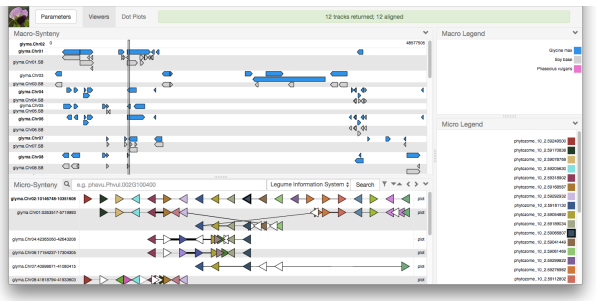

Fig. 2. Comparison of results from MCScanX using gene family assignments (blue blocks) to those produced using DAGChainer on CDS pairwise matches (gray blocks). As can be seen, the results are generally similar, with the gene-family based use of MCScanX tending to coalesce into larger blocks some regions called as fragmentary by DAGChainer. It is also clear that the GCV algorithm for determining microsynteny to the region of soybean chromosome 2 used as the query is producing results of comparable sensitivity to those produced by MCScanX, as evidenced by the small blocks from chromosomes 4 and 6 found in both the MCScanX macro-synteny blocks and the GCV microsynteny representation, but missing from the DAGChainer-based results. On the other hand, MCScanX tends to ignore inverted segments that disrupt larger collinear blocks, as displayed in the focus region for the example and present in the corresponding DAGChainer blocks- see Figure 3 for the effects of this behavior in the context of the PGDD implementation of block search and display.

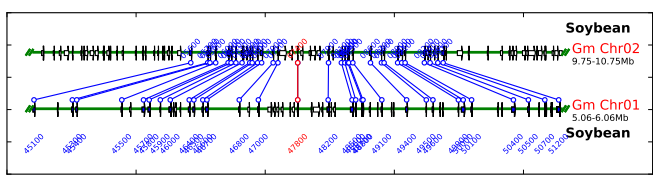

Fig. 3. The soybean chromosomes 1 and 2 comparison from region showing the same inverted segment in Figure 2, produced using the PGDD locus search. Noteworthy is that the inverted block displayed in GCV using its repeat and track merging algorithm shows up as an unmatched set of anchors (with the exception of the central gene in the inverted segment). This has the result that the choice of any of the genes within the interval as the locus to be searched fails to find the two block in which they are embedded as being syntenic. GCV's approach finds the blocks as candidates due simply to their similar gene content and initially ignores the ordering which has been disrupted by the structural variation, then applies the client-side modified alignment to detect the presence of the inversion and display accordingly.

that are predetermined to contain syntenic content. The Gobe viewer for GEvo output is implemented in Flash, and unlike the gene family strategy taken by GCV, emphasizes comparisons of HSP between pairs of sequences and their relationship to gene structural elements (e.g. exon structure), making the view more detailed than GCV and hence a little more complex to interpret when large numbers of comparisons are in play, which could obscure the identification of structural events such as copy number variation and presence/absence variation among gene clusters, although it makes it possible to see sequence-level events below the level of those that impact annotation, which would be invisible to GCV. The ability to perform on-demand sequence comparisons for genomic segments of interest provides a naturally complementary functionality to the approach taken by GCV and we are developing approaches for utilizing CoGe web services as optional plugins and linkouts through the service framework used by GCV.

Genomicus' PhyloView is similar to GCV in that it compares sequences of genes based on their gene family content. In fact, the alignment view is built from a tree representing the phylogenetic relationships among the returned tracks, where each track is displayed alongside the node it represents in the tree. This is an excellent feature and

well suited to the ancestral reconstruction of genomic segments - a key strength of the system. However, for the purposes of federating data across providers, a tight coupling of tracks even with a predetermined tree makes integration with multiple data sources non-trivial. Additionally, rather than compressing, misaligning, or inverting inexact matches, Genomicus simply omits content from other tracks not matching a gene family in the query. This could prevent the user from identifying interesting structure present in many tracks other than the query.

The feature that most fundamentally distinguishes the GCV from these tools is that the only requirement is that each genome's annotations have their gene family assignments pre-computed, which can be done independently on each genome assuming that the family definitions are sufficiently broad to capture most gene content of interest within the taxonomic group. This allows GCV to easily support data federation, only requiring that all service providers have come to agreement on a common set of gene family definitions. This model enables the user to recognize events like copy-number variation and presence/absence variation across large sets of genomic segments from taxonomic groups that span multiple genome data providers.

References

Bentley, J. L. (1975). Survey of techniques for fixed radius near neighbor searching. Technical report, Stanford Linear Accelerator Center, Calif.(USA).

Cho, I.-H., Staton, M., Lee, T., Main, D., Sanderson, L.-A., Bett, K. E., Jung, S., Cheng, C.-H., and Ficklin, S. P. (2012). Tripal: a construction toolkit for online genome databases.

Dash, S., Campbell, J. D., Cannon, E. K., Cleary, A. M., Huang, W., Kalberer, S. R., Karingula, V., Rice, A. G., Singh, J., Umale, P. E., *et al.* (2016). Legume information system (legumeinfo. org): a key component of a set of federated data resources for the legume family. *Nucleic acids research*, **44**(D1), D1181–D1188.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press.

Goodstein, D. M., Shu, S., Howson, R., Neupane, R., Hayes, R. D., Fazo, J., Mitros, T., Dirks, W., Hellsten, U., Putnam, N., *et al.* (2012). Phytozome: a comparative platform for green plant genomics. *Nucleic acids research*, **40**(D1), D1178–D1186.

Haas, B. J., Delcher, A. L., Wortman, J. R., and Salzberg, S. L. (2004). Dagchainer: a tool for mining segmental genome duplications and synteny. *Bioinformatics*, **20**(18), 3643–3646.

Kleinberg, J. and Tardos, E. (2006). *Algorithm design*. Pearson Education India.

Lee, T.-H., Kim, J., Robertson, J. S., and Paterson, A. H. (2017). Plant genome duplication database. *Plant Genomics Databases: Methods and Protocols*, pages 267–277.

Lopez, M. D. and Samuelsson, T. (2011). egob: eukaryotic gene order browser. *Bioinformatics*, **27**(8), 1150–1151.

Louis, A., Nguyen, N. T. T., Muffato, M., and Crollius, H. R. (2015). Genomicus update 2015: Karyoview and matrixview provide a genome-wide perspective to multispecies comparative genomics. *Nucleic acids research*, **43**(D1), D682–D689.

Lyons, E., Pedersen, B., Kane, J., Alam, M., Ming, R., Tang, H., Wang, X., Bowers, J., Paterson, A., Lisch, D., *et al.* (2008). Finding and comparing syntenic regions among arabidopsis and the outgroups papaya, poplar, and grape: Coge with rosids. *Plant physiology*, **148**(4), 1772–1781.

Mungall, C. J., Emmert, D. B., Consortium, F., *et al.* (2007). A chado case study: an ontology-based modular schema for representing genome-associated biological information. *Bioinformatics*, **23**(13), i337–i346.

Smith, R. N., Aleksic, J., Butano, D., Carr, A., Contrino, S., Hu, F., Lyne, M., Lyne, R., Kalderimis, A., Rutherford, K., *et al.* (2012). Intermine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data. *Bioinformatics*, **28**(23), 3163–3165.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, **147**(1), 195–197.

Wang, Y., Tang, H., DeBarry, J. D., Tan, X., Li, J., Wang, X., Lee, T.-h., Jin, H., Marler, B., Guo, H., Kissinger, J. C., and Paterson, A. H. (2012). Mcscanx: a toolkit for detection and evolutionary analysis of gene synteny and collinearity. *Nucleic Acids Research*, **40**(7), e49.