AEGIS: an annotation extraction and genomic integration resource

David Navarro-Payá[1*], Antonio Santiago[1*], Amandine Velt[2], Marco Moretto[3], Camille Rustenholz[2], José Tomás Matus[1+]

[1] Institute for Integrative Systems Biology (I[2]SysBio, UV-CSIC), Valencia, Spain

[2] INRAE-Université de Strasbourg, SVQV, 68000 Colmar, France

[3] Unit of Digital Agriculture, Research and Innovation Centre, Fondazione Edmund Mach Via E. Mach 1, 38098 San Michele all'Adige, Italy

[*] **First authors**

[+] **Corresponding author**

Abstract

The GTF/GFF3 formats are the standard for storing and exchanging genome annotations. However, their flexibility often results in inconsistent and poorly formatted files across different sources, creating a major bottleneck for downstream bioinformatics analyses. Here, we present Annotation Extraction and Genomic Integration Suite (AEGIS), a comprehensive and user-friendly command-line toolkit designed to parse, validate and standardise genome annotation files. AEGIS robustly corrects common structural and formatting errors, ensuring interoperability with downstream tools. Beyond standardisation, the suite provides advanced modules for analysis, such as flexible sequence extraction (e.g. genes, CDS, proteins) with isoform handling, customisable promoter region definitions and targeted DNA motif searches. A key feature of AEGIS is its integrated workflow for comparative genomics, which combines multiple lines of evidence (i.e., sequence homology, synteny and coordinate-based lift-overs) to enable a robust gene ID correspondence and orthology assessment. We demonstrate the utility of AEGIS by comparing two major *Arabidopsis thaliana* annotations (TAIR10 vs. Araport11), successfully identifying and quantifying complex structural changes such as gene splits and fusions. AEGIS provides a unified solution for annotation quality control, feature extraction and comparative genomic analysis, simplifying complex workflows and enhancing reliability in bioinformatic research. The software is open-source,

implemented in Python and is available on GitHub, PyPI, and as a Docker container to ensure accessibility and reproducibility.

Introduction

The efficient representation and description of genomic feature annotations are fundamental requirements in modern bioinformatics. Standardised and reliable genome annotations are essential to accurately interpret gene evolution, expression, and regulation across species. While several related specifications exist, such as the more restrictive Gene Transfer Format (GTF), the General Feature Format version 3 (GFF3) has become the standard for genome annotation exchange since its release in 2004, underpinning widely used annotation pipelines, such as MAKER (Cantarel et al., 2008) and BREAKER (Gabriel et al., 2024), as well as genome browsers such as IGV (Robinson et al., 2011) and JBrowse (Buels et al., 2016). This format has been integrated into most tools requiring genome annotation data as input. It consists of nine tab-delimited columns, with the ninth column, 'attributes', containing a semicolon-separated list of tag-value pairs. These define the relationships between features, for example, by linking exons to a parent transcript using ‘ID’ and ‘Parent’ tags in a hierarchical structure representing the complex organisation of genes and their products.

Despite the existence of a formal specification, the flexibility of the GFF3 format has led to a lack of strict adherence, which has become a significant obstacle for standardised downstream processing. Annotation files vary across organisms, frequently exhibiting inconsistencies due to the use of different source software packages or notation criteria. These discrepancies can range from simple formatting errors to more complex structural problems, hindering fundamental tasks like extracting coding sequences (CDS) for translation, calculating basic annotation statistics, or isolating promoter regions for the inspection of regulatory elements. Common issues include missing features, incorrect or inconsistent feature coordinates, improper strand orientation and incorrectly calculated phase information (in the case of coding sequences). While certain tags like 'ID' and 'Parent' have predefined meanings, different annotation providers often use non-standard tags or model the same data with different notation styles. For instance, the hierarchical relationship between an exon and its parent gene can be represented differently

across major databases like Ensembl, RefSeq and GENCODE, complicating the parsing process required to trace a feature's lineage. This variability often results in files where child features reference parent IDs that do not exist, leading to parsing errors in downstream applications. Many analysis tools rely on a consistent GFF structure as input, and inconsistent or poorly formatted files can lead to a variety of errors, from outright tool failure to more subtle and potentially misleading misinterpretations of the data. Such inconsistencies force users to develop *ad hoc*, and often brittle, scripts tailored to specific file sources.

Here, we introduce AEGIS (Annotation Extraction and Genomic Integration Suite), a tool capable of parsing and standardising GFF files from various sources to ensure their interoperability and the reliability of subsequent bioinformatic analyses. AEGIS's dual-layer design provides a powerful command-line interface (CLI) for end-users, while also allowing the use of its functionalities as a versatile Python library, making it also a foundational component for developers building custom analysis pipelines or other bioinformatics software that requires reliable handling of genomic annotations.

To showcase AEGIS's capabilities, we have developed a range of different CLI modules for downstream analyses, including simple tasks such as calculating annotation statistics and extracting feature sequences, comparable to functionalities found in AGAT (Dainat, 2022), gffread (Pertea & Pertea, 2020), or gffutils, but also more complex comparative genomics studies. The toolkit integrates workflows for analysing gene correspondence between annotation versions in the same assembly, and for assessing orthology by combining gene model overlaps via tools like Liftoff and LiftOn with additional approaches such as reciprocal BLAST searches, OrthoFinder clustering and synteny-based analyses with MCscan, thus creating a unified solution for annotation quality control and biological inquiry.

Implementation

The AEGIS toolkit comprises a suite of modular CLI tools designed to address distinct tasks in genome annotation processing and analysis. Its broad operating system (OS) compatibility is due to its full Python implementation. Each tool command follows the ***aegis {tool}*** format and can be called from a terminal as long as installation requirements are satisfied. All of the CLI tools make use of a set of purpose-built Python classes (Supplementary Figure S1) providing a solid framework for custom-made genome annotation functionalities as well as facilitating future AEGIS updates. To ensure the long-term reliability of these functionalities, an extensive suite of unit tests have been implemented covering core parser functions and CLI operations. This framework is integrated into a continuous integration pipeline via GitHub Actions, which validates every commit to the repository and is transparently documented in the package's GitHub history.

While the majority of AEGIS modules such as those for file standardization (tidy, reformat), feature extraction (extract), and coordinate-based overlap analysis (overlap), are implemented as native Python functions, the orthology command requires the local installation of several external bioinformatic tools: DIAMOND for sequence homology, MCscan (from the JCVI toolkit) for synteny analysis, OrthoFinder for orthogroup inference, and Liftoff/LiftOn for annotation transfer. Due to the complex installation of these dependencies and the need for a specific fork of LiftOn to ensure full compatibility with the AEGIS workflow, we provide a pre-configured Docker image that encapsulates the entire environment, ensuring reproducibility without the need for manual dependency management. Furthermore, we have established an automated release workflow using GitHub Actions. Upon every new release, the system automatically publishes the aegis-bio package to the Python Package Index (PyPI). Simultaneously, a new Docker image is built from the repository’s Dockerfile and uploaded to both Docker Hub and the GitHub Container Registry (GHCR), ensuring that users always have access to synchronised versions across all platforms.

To systematically present the breadth of the current capabilities (Figure 1), we first describe the core utilities for foundational file manipulation, standardisation and data integration. Subsequently, we detail the modules for annotation exploration, which

enable quantitative summarisation and flexible data subsetting. Finally, we present the advanced analytical modules designed for downstream biological discovery, including tools for comparative genomics and annotation comparison.

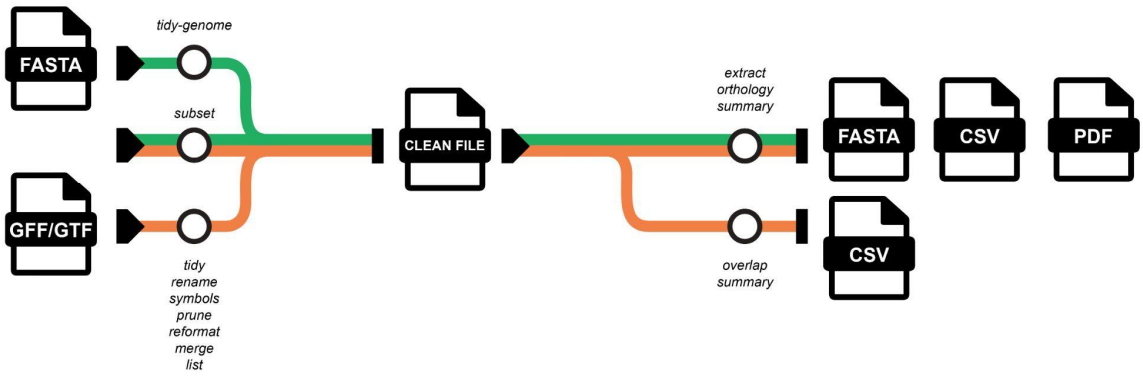


Figure 1: Overview of AEGIS package commands. A range of commands require annotation file(s) as input, while others simply require the assembly files. Functionalities such as sequence extraction and orthology analysis require both genome annotation and assembly file(s). The command '*aegis --help*' displays a general functional description for every tool.

<u>Preprocessing and standardisation of genome annotation and assembly files</u>

AEGIS has been equipped with a set of general tools to deal with common preprocessing needs and further downstream analyses. These tools provide a flexible framework for managing and standardising genome annotations for a variety of downstream applications.

The ***reformat*** tool converts annotation files between GTF and GFF formats to ensure compatibility with external software. All AEGIS tools are GTF and GFF compatible so inputs can interchangeably follow either file format (i.e. the input format is automatically determined based on file content).

The ***tidy*** command has been implemented as a standardising and error-correcting tool that can generate features if they do not exist (e.g. untranslated regions or exons). It can also polish parent-referencing, feature coordinates and phase values, while providing a set of options to customise output GFFs such as filtering for a

143 particular biotype or whether to include untranslated regions (UTRs) in the output
144 file. Whilst the standard exon format style for a GFF should involve the use of the
145 'parent' attributes wherever transcripts share the same exon, some GFF styles add
146 all exon entries below each transcript regardless of repetition. AEGIS can accept
147 both input formats and the ***tidy*** command can be used to choose which output style
148 is chosen for exons, in order to fulfil external tool requirements. Similarly, parameters
149 to choose the style of CDS entry ids are also provided.

150 The ***merge*** tool can combine any number of annotation files together whilst
151 preventing id clashes and offering control over redundancy removal within the same
152 loci through coordinate overlap parameters. Conversely, to reduce annotation size,
153 AEGIS offers ***prune*** and ***subset***. ***Prune*** accepts lists of ids to remove, transcript or
154 gene based, and solves derived parent and subfeature issues. ***Subset*** has been
155 designed to help testing and debugging efforts by generating custom, lighter
156 versions of annotation files with corresponding subsetting of their FASTA formatted
157 genome, thus reducing unnecessary workloads.

158 The ***rename*** tool provides highly customisable modification of feature identifiers
159 (gene, transcript, etc.). This includes removing or adding prefixes/suffixes,
160 standardising transcript ID styles, or configuring new identifier structures for updated
161 releases. In addition, the ***symbols*** tool allows users to append or replace gene
162 symbols within an annotation file.

163 The ***tidy-genome*** tool provides similar functionality to ***rename*** and ***tidy***, but it is
164 applied to genome assembly files. Non-chromosomal level scaffolds or organelles
165 can be selectively removed. Moreover, it allows FASTA feature renaming and
166 optional synchronisation of changes in a matching annotation file.

167 <u>Annotation summary report and targeted feature extraction</u>

168 The **list** tool provides a convenient computational approach to generate tabular
169 summaries of genes or transcripts from an annotation file, exporting the resulting
170 datasets into TSV or CSV formats. Users can extensively customise the output
171 tables by appending various metadata attributes, such as feature lengths, genomic
172 coordinates, chromosome assignments, coding status, and gene symbols.

173 Additionally, the tool includes flexible filtering parameters to selectively retain or
174 exclude specific feature classes, such as coding or non-coding elements,
175 pseudogenes, and transposable elements.

176 The ***extract*** command can output multiple types of genomic features, including
177 whole genes, CDSs, their translated protein sequences and putative promoter
178 regions. The handling of RNA features takes a more sophisticated approach allowing
179 precise export of all annotated transcripts or filtering for specific biotypes. For
180 example, users can isolate only messenger RNAs (mRNAs) for expression
181 quantification from poly(A)-selected libraries (e.g. to be used with pseudoaligners like
182 Salmon or Kallisto), or they can extract specific classes of non-coding RNAs
183 (ncRNAs), such as long non-coding RNAs (lncRNAs) or microRNAs (miRNAs), for
184 studies focusing on non-coding gene function.

185 Beyond custom feature selection, users can control the inclusion of isoforms through
186 distinct extraction modes (Figure 2). For instance, it can retrieve either all annotated
187 variants for a given gene (“all” mode) or only a single, canonical representative,
188 defined by default as the longest isoform (“main” mode). This is particularly useful for
189 generating non-redundant datasets for comparative genomics or functional
190 annotation. Furthermore, the tool provides specialised parameters to generate
191 unique sets of protein or CDS sequences, either on a per-gene basis (to study
192 protein diversity from a single locus) or across the entire genome (to create a non-
193 redundant proteome database).

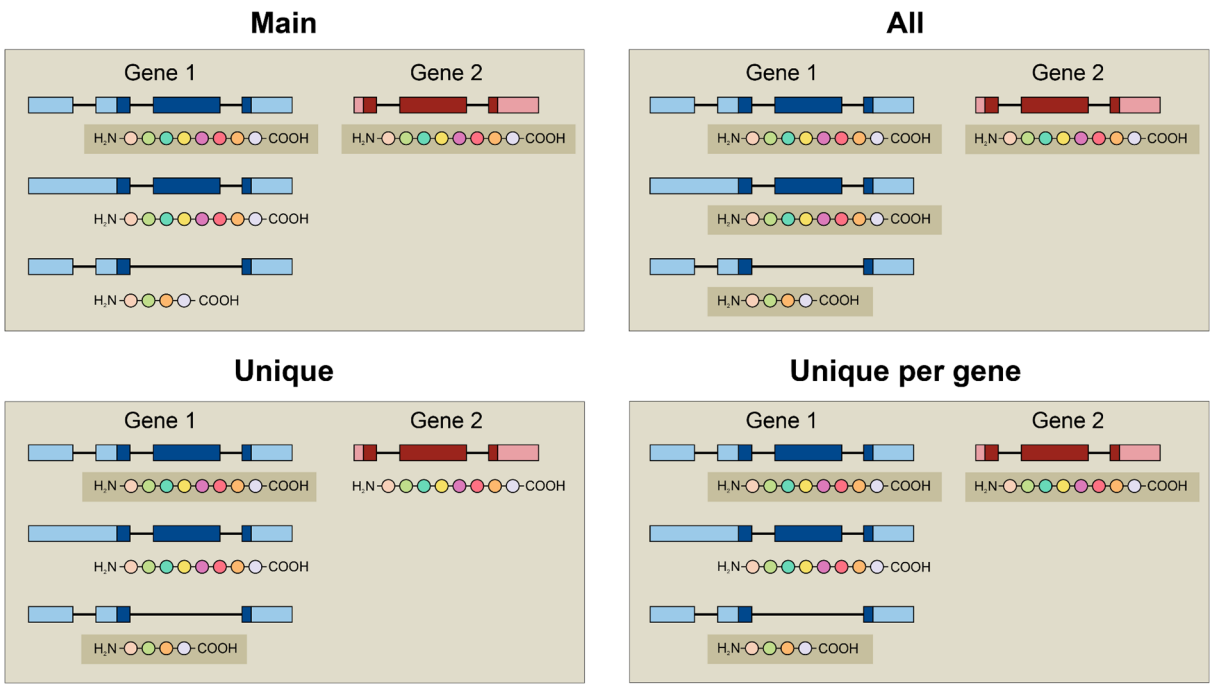


Figure 2. Schematic representation of the sequence extraction modes in *extract*. The figure illustrates the four modes available, via the '*--mode*' argument, to control the filtering of transcript and protein sequences. The example depicts two genes: gene 1 produces two distinct protein isoforms from its three transcripts, while gene 2 produces the same protein isoform as gene 1 from one transcript. "Main" selects a primary transcript for each gene (defined by default as the one with the longest CDS); "All" extracts all annotated protein products for each gene, without any filtering; "Unique" applies a global filter, extracting only one representative unique protein sequence found across the entire dataset, effectively removing all protein redundancy within and between genes; "Unique per gene", identifies all unique protein isoforms for each gene. Coloured boxes represent exons (CDS darker, UTRs lighter) and lines represent introns. The shaded background highlights the protein sequence output for each mode.

In the case of promoter extraction, users are not limited to a standard definition of a fixed-length region upstream of the transcription start site (TSS). The tool also allows for the extraction of regions upstream of the translation start codon (ATG), and a hybrid option which extracts the region upstream of the TSS plus the entire 5' UTR up to the ATG (in case it is annotated). These options are particularly useful in the case of studying regulatory elements when UTRs are poorly annotated (Figure 3).

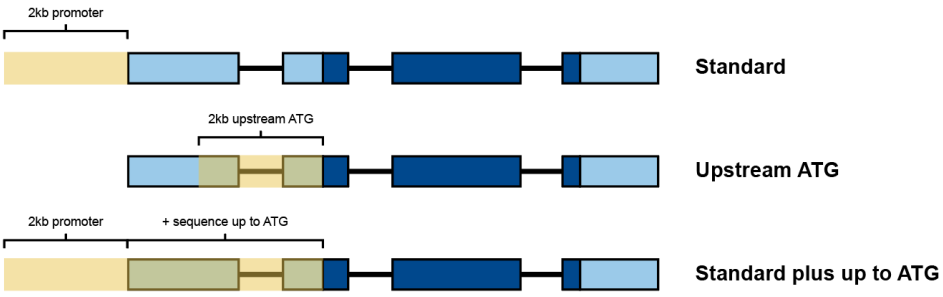


Figure 3. Illustration of promoter extraction strategies implemented in *extract*. Three different modes for promoter definition are supported when using the '*--promoter-type*' option: (Top) "Standard": Extracts a fixed-length region (2 kb by default) upstream of the TSS. (Middle) "Upstream ATG": Extracts the region upstream of the ATG. (Bottom) "Standard plus up to ATG": Combines the standard TSS-based promoter with the additional sequence between the TSS and the ATG, thereby including potential 5' UTR elements and introns. Blue boxes represent exons (CDS darker, UTRs lighter) and lines represent introns. The highlighted yellow area depicts the extracted promoter sequence.

Finally, this tool provides extensive customisation of the output FASTA headers to facilitate data management and integration with other software. For example, users can choose to label all output sequences (e.g. transcripts, proteins) with their parent gene ID, which simplifies mapping results back to the gene level. An optional verbose output can also append genomic coordinates and other metadata directly to the sequence headers, ensuring full traceability of the extracted features.

Comparative and functional analysis of genome annotations

Detecting and quantifying feature overlaps

To identify spatial overlaps between features, the ***overlap*** command includes a method where the algorithm operates in two primary modes: (1) detecting overlaps between features within a single genome annotation file (intra-annotation), and (2) comparing features between two different annotation files (inter-annotation). A fundamental prerequisite for inter-annotation comparison is that both annotation files must share the same genome assembly, ensuring a common coordinate system. This is feasible when the annotations are different annotation versions of the same genome assembly, but for other cases, the use of external tools such as Liftoff (Shumate & Salzberg, 2021) or LiftOn (Chao et al., 2024) in advance is recommended. Alternatively, the ***orthology*** CLI described below may be more appropriate.

For each pair of potentially overlapping genes, the algorithm performs a hierarchical analysis to quantify the extent of the overlap at multiple feature levels (gene, exon and CDS). All of the calculated metrics take into account both query and target overlap percentages so that 100% is only achievable if the set of coordinates compared is exactly the same.

- Gene-level overlap: The initial overlap is calculated based on the full genomic span of the two genes (from start to end coordinates). The number of overlapping base pairs (bp) is determined and expressed as a percentage of the total length of the query gene and the target gene, respectively. The relative strand orientation (same or opposite strand) is also recorded.

$$minGeneOverlap = \min\left(\frac{overlap(g_1, g_2)}{g_{1Length}}, \frac{overlap(g_1, g_2)}{g_{2Length}}\right) \times 100$$

- Exon-level overlap: For a more in-depth analysis, the algorithm assesses overlap at the exon level. It compares all possible transcript pairs between the two genes. For each transcript pair, it calculates the cumulative length of all overlapping exonic regions. The maximum cumulative overlap value found among all transcript pairs is retained as the representative exon-level overlap. This value is then normalised against the total exon-spanned length of the two transcripts that produced this maximal overlap.

$$minTranscriptOverlap = \min\left(\frac{\max\left(overlap(g_{1Tn}, g_{2Tn}, \dots)\right)}{g_{1TnLength}}, \frac{\max\left(overlap(g_{1Tn}, g_{2Tn}, \dots)\right)}{g_{2TnLength}}\right) \times 100$$

- CDS-level overlap: A similar procedure is applied to the CDSs. The algorithm identifies the transcript pair that yields the maximum cumulative overlap between their respective CDS segments. This value is subsequently expressed as a percentage of the total CDS length for each of the corresponding transcripts.

$$minCDSOverlap = \min\left(\frac{\max\left(overlap(g_{1Cn}, g_{2Cn}, \dots)\right)}{g_{1CnLength}}, \frac{\max\left(overlap(g_{1Cn}, g_{2Cn}, \dots)\right)}{g_{2CnLength}}\right) \times 100$$

To systematically assess the structural conservation between query gene annotations and their mapped targets, we developed a custom hierarchical scoring

system (Supplementary Table S1). This system quantifies the degree of overlap by prioritising the most functionally significant regions of a gene. For each query-target pair, the algorithm first attempts to calculate the minimum reciprocal overlap at CDS level. If a pair does not share overlapping CDS features (or if one of the genes is non-coding), the comparison proceeds to evaluate overlap at the exon level. If no exon overlap is found, the system finally assesses the overlap of the full gene loci (this only happens with intron-nested genes or genes with no annotated transcripts). This hierarchical approach yields a single overlap score from 0 to 11, where higher scores reflect greater structural identity (e.g. a score of 11 depicts a 100% reciprocal CDS overlap).

The algorithm also integrates pre-existing synteny information for inter-annotation comparisons, should the original annotation file, for a lifted over or transferred annotation, be provided. If a gene involved in an overlap maintains the same immediate neighbouring genes after lift-over, this status is recorded along with the overlap data. All gene overlaps can be exported in tabular format for manual inspection.

Orthology

Whilst ***overlap*** focuses on comparing annotation files associated with the same genome, ***orthology*** is designed to extensively compare annotations associated with different genomes. The outputs of multiple tools (internal and external) are integrated into an automated pipeline designed to generate a set of orthologues sorted into several confidence tiers.

The approach combines four complementary lines of evidence:

1. Sequence homology: the fast protein aligner DIAMOND performs reciprocal sequence searches (Buchfink et al., 2015), processing the results to identify BLAST results above a certain coverage (≥30% by default) and an e-value (≤1×10^{-5} by default) threshold. These hits are then used to determine reciprocal best hits (RBHs), reciprocal best BLAST hits (RBBHs) and the remaining results which will be only forward (fw) or reverse (rv) results depending on which annotation is considered the ‘query’. An additional

identity (≥30% by default) filter for single BLAST hits (fw or rv) is also used. The threshold values are fully adjustable.

2. Synteny and collinearity: to incorporate genomic context, AEGIS uses the JCVI toolkit's MCscan modules (Tang et al., 2024). By analysing the relative order of genes between two genomes, it identifies conserved syntenic blocks. Orthologue pairs found within these blocks receive additional support, as their relationship is preserved by both sequence similarity and genomic location.
3. Annotation lift-over and overlap analysis: AEGIS integrates annotation transfer tools like Liftoff (Shumate & Salzberg, 2021) and LiftOn (Chao et al., 2024) to project gene models from a source genome to a target genome. Uniquely, AEGIS then applies its native spatial overlap algorithm to the transferred annotations and filters overlap results using an ***overlap*** score threshold of ≥6.
4. Multi-species orthogroup inference: To incorporate phylogenetic and evolutionary information, AEGIS performs a full OrthoFinder analysis (Emms & Kelly, 2019). It extracts the primary proteome for each input species annotation, runs the OrthoFinder pipeline to cluster genes into orthogroups, and integrates the resulting pairwise orthologues from the OrthoFinder analysis into its final summary.

As output, AEGIS consolidates all sources of evidence into a single and easy-to-parse summary table with a confidence score for each putative orthologue pair. The scoring is hierarchical, where pairs classified as high confidence are primarily defined by strong evidence from lift-over based mapping, requiring a positional overlap score of ≥ 7 from either Liftoff or LiftOn (Supplementary Table S2). Pairs lacking this positional evidence are designated as medium confidence, provided they are supported by the combination of being an RBH and part of a syntenic block identified by MCscan, as well as pairs identified as RBBH alone. All other pairs identified through less stringent criteria are categorised as lower confidence, involving relationships supported by only one of the primary methods (e.g. an RBH without syntenic confirmation).

Results

Comparative analysis of consistency and divergence in Arabidopsis genome annotations

To showcase AEGIS's capabilities in comparative genomics, firstly for evaluating the consistency and evolution of gene models across annotation versions within the same genome, we performed an intra-species overlap analysis between TAIR10 and Araport11 (Cheng et al., 2017; Lamesch et al., 2012). Both annotation sets are built on the latest *Arabidopsis thaliana* TAIR10 genome assembly, allowing direct comparison of feature coordinates. Using the ***overlap*** tool, we quantified the similarity between gene models in these two major *A. thaliana* annotations (Supplementary Table S3). This analysis revealed genes with conserved structural definitions across versions, as well as those showing substantial changes in their models, including structural modifications and, in some cases, changes in gene identifiers.

The initial gene-level overlap analysis revealed a high degree of concordance for the majority of Araport11 genes (31,733 with a ≥10 overlap score). This is expected for two annotations of the same genome, where genes retain their original identifier in both TAIR10 and Araport11 but where some of them may show differences in their genomic span or exonic structure. Conversely, the overlap analysis also captured cases where a single gene from the older TAIR10 annotation splits into multiple gene models in Araport11. This fragmentation often indicates a re-evaluation of gene boundaries based on improved prediction algorithms or the incorporation of additional transcriptomic evidence, where a previously annotated single locus is now recognised as two or more distinct genes. For instance, the gene ID *AT2G52240* in TAIR10 was found to have a partial overlap with *AT2G52240* and *AT2G52245* in Araport11, with the latter being a small transcript variant coming from the previous model (Figure 4A). The opposite scenario (i.e., gene merge cases), was also observed, where two or more distinct TAIR10 genes were reported to overlap significantly with a single, larger Araport11 gene (e.g. *AT1G27595*) (Figure 4B). This analysis allows users to track structural gene model changes over annotation updates, which are not always provided in such a level of detail.

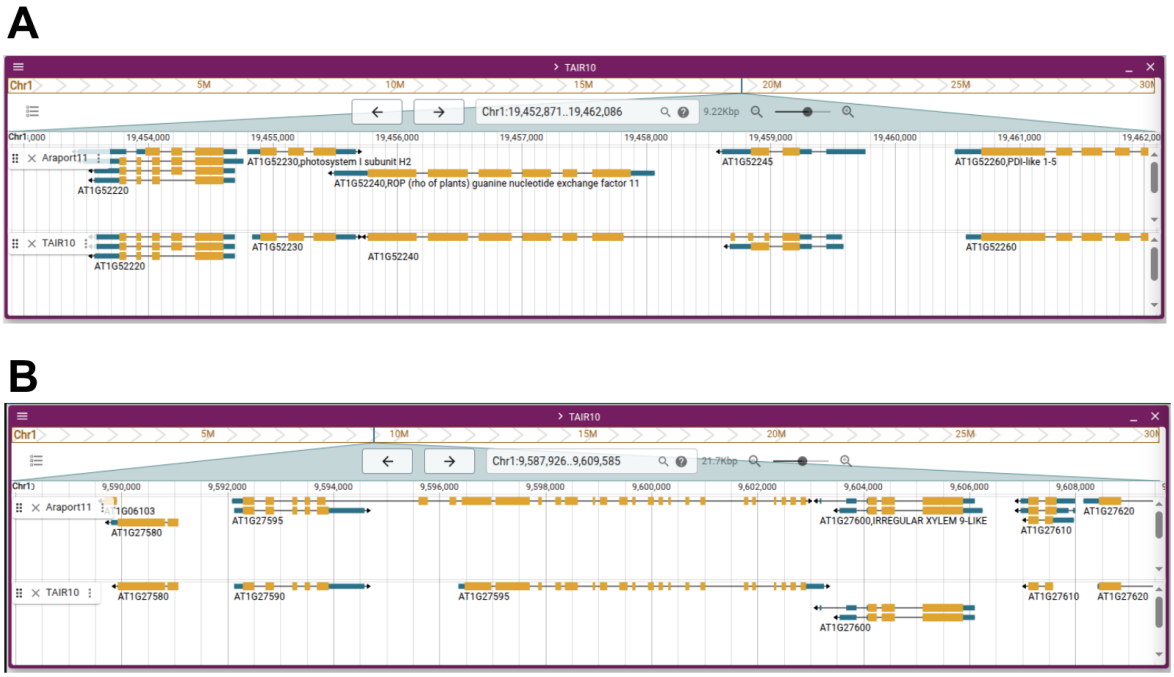


Figure 4. Annotation version differences as detected by *overlap*. Views of different regions in chromosome 1 in Jbrowse2, with examples of gene model changes between Araport11 (top track in each panel) and TAIR10 (bottom track) structural annotations (Arabidopsis thaliana TAIR10 genome assembly). **(A)** Example of a gene model split. The TAIR10 annotation identified a single long gene, *AT1G52240*, whereas Araport11 resolved this as two distinct genes: a redefined and shorter *AT1G52240*, and a new gene model; *AT1G52245*. **(B)** Example of a gene model merge. In the TAIR10 annotation, this locus contained two separate genes, *AT1G27590* and *AT1G27595*. The updated Araport11 annotation merged these into a single, longer gene model, *AT1G27595*. In both panels, boxes represent exons (CDSs in yellow and UTRs in blue) and black horizontal lines represent introns.

Orthology between grapevine, tomato and Arabidopsis genomes

Furthermore, to demonstrate the utility of AEGIS in an inter-species context, we compared genome annotations from *Arabidopsis thaliana*, tomato (*Solanum lycopersicum*) and grapevine (*Vitis vinifera*). This cross-species comparison enabled the identification of conserved orthologues and highlighted structural differences in their gene models.

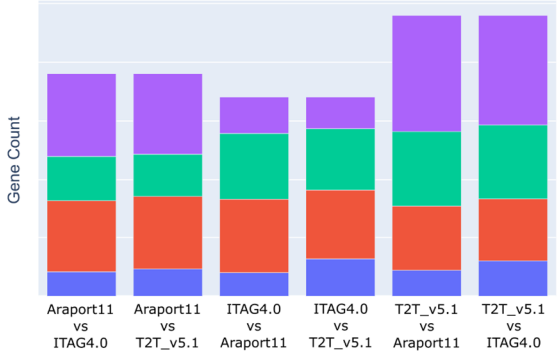
The annotations and assemblies of grapevine (annotation v5.1, assembly T2T.v5; Shi et al., 2023), tomato (annotation ITAG4.0, assembly ITAG4; Hosmani et al., 2019) and Arabidopsis (annotation Araport11, assembly TAIR10; Lamesch et al., 2012) were compared with ***orthology*** resulting in a list of associated gene ID pairs

381 between annotations, supporting evidence details and a qualitative summary score.
382 This process resulted in a comprehensive list of orthologous gene pairs, each with all
383 of the supporting evidence details and a qualitative summary score (Figure 5A).
384 More than half of the genes of each genome, between the three species compared,
385 had at least one orthologue in each pairwise comparison. For instance, 23,978
386 Arabidopsis genes had at least one orthologue in tomato when looking at the
387 combined results from all tools. This combined value is higher than any of the
388 individual metrics, supporting the benefit of the approach (Figure 5B). The
389 Arabidopsis (Araport11) and tomato (ITAG4.0) comparison shows a degree of
390 overlap amongst individual ***orthology*** tool results (Figure 5C). The greatest set
391 (5,411 genes) corresponds to the shared findings of MCscan ('anchors' and
392 'last_filtered'), RBBH and OrthoFinder, suggesting that the output of these tools is
393 highly concordant. Nevertheless, individual tool results also manage to uniquely
394 detect orthologues for a considerable number of genes (e.g. 2,360 genes are
395 singularly detected by RBHs), demonstrating that each tool can contribute to the final
396 combined result.

397 The output table with the default ***orthology*** command shown in Figure 5 results in
398 over 900,000 orthologue gene-pairs. The same tool can be used with a different
399 parameter to produce a lighter output (Supplementary Table S4). In this case,
400 ~125,000 orthologue gene-pairs were identified, which still manages to find
401 orthologue matches for a sizeable portion of annotated genes (e.g. more than 50%
402 for both Arabidopsis and tomato). This drop mainly affects the lower-confidence
403 orthologues (Supplementary Figure S2A) and there is still a benefit in combining the
404 different results of each tool (Supplementary Figure S2B).

A

RBHs and Fw/Rv Blasts Included



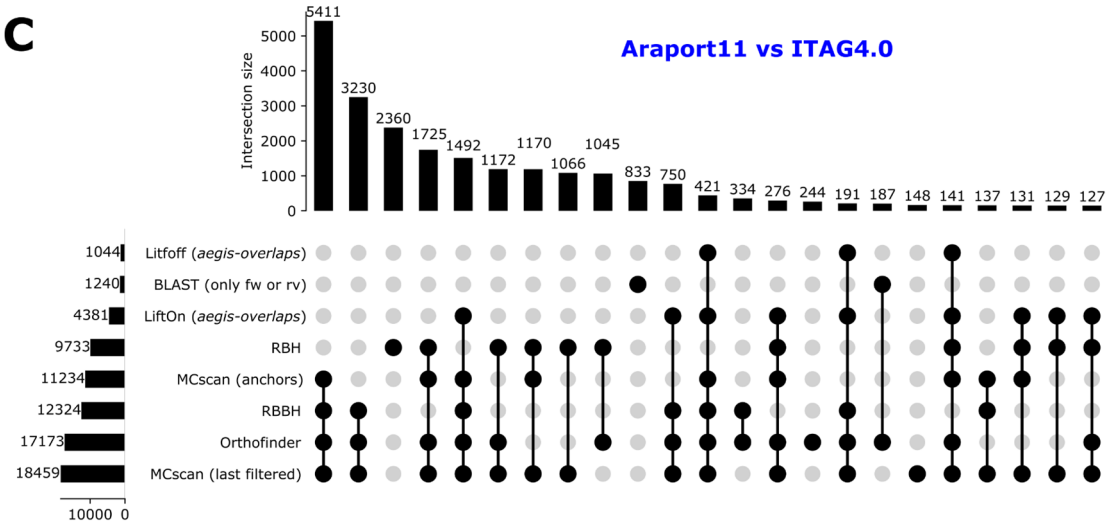
gene_id_A	gene_id_B	score	summary_score
AT1G01170	Solyc10g080190.1 (ITAG4.1, tomato)	**rev_lifton_aegis** [10], **mcscan_last_filtered** [137], **Rbbh** [evalue=1.06e-37/6.57e-38, norm_bitscore=1.4/1.6, coverage=86.9/94.8, identity=69.9/69.9], **orthofinder** [OG0002865]	high
AT1G01170	Vitvi05_01chr08g23980 (T2T_v5.1, grapevine)	**mcscan_last_filtered** [157], **rbbh** [evalue=2.38e-34/1.1e-34, norm_bitscore=1.5/0.1, coverage=90.5/8.9, identity=73.7/73.7], **orthofinder** [OG0002865]	medium
AT1G01170	Solyc04g150111.1 (ITAG4.1, tomato)	**mcscan_last_filtered** [124], **orthofinder** [OG0002865]	lower

Orthology Summary Score
High, Lower, Meidium, None

B

Comparison	Litfoff (aegis-overlaps)	BLAST (only fw or rv)	LitfOn (aegis-overlaps)	RBH	MCscan (anchors)	RBBH	OrthoFinder	MCscan (last filtered)	None	*Aegis-orthology* (all tools combined)
Araport11_ITAG4_0	1,044	1,240	4,381	9,733	11,234	12,324	17,173	18,459	14,112	**23,978**
Araport11_T2T_v5_1	1,303	1,292	4,971	9,587	12,775	12,688	17,446	18,860	13,750	**24,340**
ITAG4_0_Araport11	1,127	3,953	4,193	10,741	10,674	12,324	17,872	18,034	6,184	**27,891**
ITAG4_0_T2T_v5_1	3,411	3,538	8,565	10,940	13,526	13,371	19,701	19,714	5,380	**28,695**
T2T_v5_1_Araport11	1,306	4,170	4,707	9,365	10,561	12,688	18,127	16,479	19,765	**28,206**
T2T_v5_1_ITAG4_0	3,308	4,016	8,185	10,065	12,014	13,371	20,327	17,720	18,645	**29,326**

C



405

Figure 5. Overview and performance of the *orthology* tool. Used command: "*aegis orthology Araport11_GFF3_genes_transposons.20250813.gff ITAG4.0.gff3 5.1_on_T2T_ref.gff3 –annotation-names Araport11,ITAG4_0,T2T_v5_1 --group-names Arabidopsis,Tomato,Grapevine –genome-files TAIR10_genome.fasta,S_lycopersicum_chromosomes.4.00.fa,T2T_ref.fasta*". The reported counts represent the number of genes in a query annotation that have at least one orthologue identified in the target annotation. **(A)** Summary of identified orthologues for Arabidopsis, tomato, and grapevine genes. The stacked bar chart (left) shows, for each annotation, how its genes are distributed according to whether an orthologue is present in the target species and, if so, the confidence of the best-supported orthologue. These qualitative levels are based on the combined results of individual tools, as shown for gene *AT1G01170* on the right. **(B)** Orthologue counts per tool result for each query annotation when compared to different target species. The combined results represent an improvement over individual tools in the number of genes with at least one orthologue. **(C)** UpSet plot illustrates the intersections of orthologue predictions among different methods for the Arabidopsis (Araport11) versus tomato (ITAG4.0) comparison. The vertical bars represent the number of genes with detected orthologues in each intersection (indicated by connected dots), while the horizontal bars on the left show the total number of genes with at least one orthologue identified by each method.

Computational performance and benchmarking

To evaluate the computational efficiency of AEGIS, we benchmarked its core utilities against comparable functionalities in an existing state-of-the-art toolkit, specifically AGAT. Benchmarks were executed on a workstation equipped with an Intel Core i9-10900 and 32 GB, recording both total execution time and memory usage (RAM).

To ensure reproducibility and consistent execution environments, all evaluated tools were containerised and executed using Docker. Upon the initialization of a tool's container, the monitoring script continuously polled the Docker Engine API via the native docker stats command to capture real-time memory usage. Data points were recorded at a sampling interval of 0.1 seconds throughout the entire lifecycle of the containerised process. The resulting telemetry data was logged as a time-series dataset.

The comparative analysis was performed using small (*Chlamydomonas reinhardtii*), medium (*Arabidopsis thaliana*), and large genomes (*Homo sapiens*). We evaluated performance across three standard bioinformatic tasks: annotation parsing and standardisation, conversion to GTF, and protein extraction.

Across all tested benchmarks and genome sizes, AEGIS demonstrated a significantly higher computational efficiency compared to AGAT (Supplementary Figure S3). In terms of execution time, AEGIS completed the tasks of GFF tidying, GTF conversion, and protein extraction around 3-6 times faster than the corresponding AGAT scripts. This performance gap was particularly evident in the *Homo sapiens* dataset, where for instance, in GTF conversion, AEGIS finished in 194 seconds compared to AGAT's 1,101 seconds.

Regarding memory management, although AEGIS reaches a peak memory usage more rapidly than AGAT, it maintains this footprint for a much shorter duration, drastically reducing the total computational cost. While AGAT showed a steady and prolonged memory plateau, AEGIS optimised resource release immediately upon task completion. These results indicate that AEGIS is highly scalable and particularly well-suited for large genomic annotations.

Discussion

Implemented in Python, AEGIS provides both a powerful command-line interface for pipeline integration and an object-oriented library for custom script development, offering a flexibility absent in tools designed for one or a few tasks. It incorporates functionalities found across several tools, including robust file parsing, format conversion, filtering, and statistical reporting, matching the feature sets of mature tools like AGAT. The analysis of genome annotation files is supported by a diverse software range, which can be broadly categorised into comprehensive command-line toolkits (e.g. AGAT, GenomeTools), specialised utilities (e.g. GffCompare, TransDecoder), and programmatic libraries (e.g. gffutils). While established toolkits provide several functionalities for genome annotation handling, this new package offers an advantage in terms of parsing capabilities, being able to process complex annotation files that cause other tools to fail. For instance, AGAT successfully resolves overlapping exons within individual transcripts, but it fails to consolidate identical exons shared across multiple transcripts. Instead, AGAT redundantly duplicates these shared features, assigning each iteration a unique ID and restricting them to a single parent. In contrast, AEGIS can optionally produce a much more compact GFF3 format by making use of the multi-parent feature of the official GFF3 specification, thereby reducing unnecessary data redundancy. Moreover, some

473 known errors in AGAT, such as the “Bio::Root::Exception” caused by long lines in
474 FASTA files when extracting feature sequences, have been avoided when designing
475 AEGIS. AEGIS is specifically engineered to handle such edge cases, with a
476 codebase designed for robustness and modular reuse. Python was selected as the
477 implementation language to maximise these attributes, favouring maintainability and
478 flexible data handling over the performance optimisations typical of lower-level
479 languages.

480 In terms of performance limitations, the AEGIS package has been benchmarked
481 favourably against AGAT for the functionalities that were compared. Nevertheless,
482 the performance speed of AEGIS or AGAT compared to other C/C^{++} tools such as
483 GFF Utilities and GenomeTools is still limited based on the chosen languages
484 themselves, Python and Perl, respectively. For basic standard operations GFF
485 Utilities and GenomeTools may still be the option of choice, however, these offer
486 less flexibility and robustness with respect to edge cases. Moreover, whilst AEGIS is
487 able to deliver GFF3 recommendations more effectively in the outputs, AGAT offers
488 greater flexibility when dealing with particular GTF format flavours.

489 AEGIS includes several high-level analytical capabilities not found in other examined
490 software (Supplementary Table S5). It provides modules for identifying a gene’s
491 main isoform, extracting a range of sequences, manipulating FASTA files together
492 with their corresponding annotation files, detecting overlaps between features, and
493 comprehensive orthology analysis. We showed that we were able to quantify and
494 categorise intra-species changes between annotation versions using the ***overlap***
495 command. It generated a detailed and up-to-date table of gene model
496 correspondences, extending beyond a static list of obsolete identifiers (such as the
497 ones provided in Araport11; (Cheng et al., 2017), where users can inspect and trace
498 gene model differences between annotation releases.

499 Moreover, AEGIS can be used to compare annotation files from different genome
500 assemblies, addressing a complex comparative orthology analysis that lies beyond
501 the scope of existing tools like GffCompare (Pertea & Pertea, 2020). While
502 GffCompare focuses on RNA processing, comparing coordinates primarily at the
503 intron and exon levels to match intron chains, AEGIS ***overlap*** is more focused on

whether an overlapping pair of genes produce a similar functional product. It calculates mutual overlap percentages at the gene, exon, and CDS levels, specifically validating that CDS segments maintain the same reading frame and retrieving exact metrics for each pairwise correspondence. The***orthology*** tool integrates the overlaps and the output of several other tools and can be used to study genome evolution and even assist in the identification of core and dispensable genes when dealing with pangenomes. We found grapevine orthologues for 24,350 tomato genes and tomato orthologues for 24,099 grapevine genes, a substantial increase over previous studies, which reported 16,454 and 15,631 orthology relationships in tomato and grapevine, respectively (Ambrosino et al., 2018). By also providing medium and lower confidence orthologues, we highlight the ability to extract more comprehensive evolutionary signals from genome annotations, providing a stronger basis for comparative genomics and functional inference.

While the ***orthology*** example use case focuses on plant genomes, we expect that this tool and the rest of the AEGIS suite will be broadly applicable to the analysis of animal and other eukaryotic genomes as well. We anticipate that AEGIS will be a valuable framework as it unifies essential manipulation tasks in genome annotation around a flexible object-oriented core from which further functionalities can be reliably developed.

Availability and requirements

AEGIS is available under the GPL-3 Open Source license. The version of the package associated with this work is distributed as a Python package in the Python Package Index as aegis-bio (https://pypi.org/project/aegis-bio/). The source code, along with installation and implementation instructions, is also available at GitHub (https://github.com/Tomsbiolab/aegis). A readme file is included for a comprehensive description of its features and their execution. AEGIS is also available as a containerised CLI along with other tools required for some functionalities, available as a Docker Image at Docker Hub (https://hub.docker.com/r/tomsbiolab/aegis). The package is OS independent.

Acknowledgements

This work was supported by the grants Valinet (PID2021-128865NB-I00) and Metacell (PID2024-163039NB-I00) awarded to J.T.M. from the Ministerio de Ciencia, Innovación y Universidades (MCIU, Spain), Agencia Estatal de Investigación (AEI, Spain /10.13039/501100011033), and Fondo Europeo de Desarrollo Regional (FEDER, European Union), and to the PROMETEO grant (PROMETEO/2021/056) awarded to J.T.M. and its associated doctoral grant (PROMETEO/2021/056-01) granted to A.S., by the Generalitat Valenciana (GVA). Further support was provided by the Grapedia COST innovators grant (IG17111) awarded to J.T.M. and C.R. This research also received a grant (IEC-BG1-2023-4), awarded to J.T.M., from the Institute for Catalan Studies (IEC) within the framework of the first call for biogenome (genomics of biodiversity) research grants. Testing and debugging was carried out on the HPC cluster Garnatxa at the Institute for Integrative Systems Biology (I^2SysBio).

References

Ambrosino, L., Ruggieri, V., Bostan, H., Miralto, M., Vitulo, N., Zouine, M., Barone, A., Bouzayen, M., Frusciante, L., Pezzotti, M., Valle, G., & Chiusano, M. L. (2018). Multilevel comparative bioinformatics to investigate evolutionary relationships and specificities in gene annotations: An example for tomato and grapevine. *BMC Bioinformatics*, *19*(S15), 435. https://doi.org/10.1186/s12859-018-2420-y

Buchfink, B., Xie, C., & Huson, D. H. (2015). Fast and sensitive protein alignment using DIAMOND. *Nature Methods*, *12*(1), 59–60. https://doi.org/10.1038/nmeth.3176

Buels, R., Yao, E., Diesh, C. M., Hayes, R. D., Munoz-Torres, M., Helt, G., Goodstein, D. M., Elsik, C. G., Lewis, S. E., Stein, L., & Holmes, I. H. (2016). JBrowse: A dynamic web platform for genome visualization and analysis. *Genome Biology*, *17*(1), 66. https://doi.org/10.1186/s13059-016-0924-1

Cantarel, B. L., Korf, I., Robb, S. M. C., Parra, G., Ross, E., Moore, B., Holt, C., Sánchez Alvarado, A., & Yandell, M. (2008). MAKER: An easy-to-use annotation pipeline designed for emerging model organism genomes. *Genome Research*, *18*(1), 188–196. https://doi.org/10.1101/gr.6743907

Chao, K.-H., Heinz, J. M., Hoh, C., Mao, A., Shumate, A., Pertea, M., & Salzberg, S. L. (2024). Combining DNA and protein alignments to improve genome annotation with LiftOn. *Genome Research*, genome;gr.279620.124v2. https://doi.org/10.1101/gr.279620.124

Cheng, C., Krishnakumar, V., Chan, A. P., Thibaud-Nissen, F., Schobel, S., & Town, C. D. (2017). Araport11: A complete reannotation of the *Arabidopsis*

thaliana reference genome. *The Plant Journal*, *89*(4), 789–804. https://doi.org/10.1111/tpj.13415

Dainat, J. (2022, January). *Another Gtf/Gff Analysis Toolkit (AGAT): Resolve interoperability issues and accomplish more with your annotations.* Plant and Animal Genome XXIX Conference, San Diego, California, United States. https://github.com/NBISweden/AGAT

Emms, D. M., & Kelly, S. (2019). OrthoFinder: Phylogenetic orthology inference for comparative genomics. *Genome Biology*, *20*(1), 238. https://doi.org/10.1186/s13059-019-1832-y

Gabriel, L., Brůna, T., Hoff, K. J., Ebel, M., Lomsadze, A., Borodovsky, M., & Stanke, M. (2024). BRAKER3: Fully automated genome annotation using RNA-seq and protein evidence with GeneMark-ETP, AUGUSTUS, and TSEBRA. *Genome Research*, *34*(5), 769–777. https://doi.org/10.1101/gr.278090.123

Hosmani, P. S., Flores-Gonzalez, M., Van De Geest, H., Maumus, F., Bakker, L. V., Schijlen, E., Van Haarst, J., Cordewener, J., Sanchez-Perez, G., Peters, S., Fei, Z., Giovannoni, J. J., Mueller, L. A., & Saha, S. (2019). *An improved de novo assembly and annotation of the tomato reference genome using single-molecule sequencing, Hi-C proximity ligation and optical maps*. Genomics. https://doi.org/10.1101/767764

Lamesch, P., Berardini, T. Z., Li, D., Swarbreck, D., Wilks, C., Sasidharan, R., Muller, R., Dreher, K., Alexander, D. L., Garcia-Hernandez, M., Karthikeyan, A. S., Lee, C. H., Nelson, W. D., Ploetz, L., Singh, S., Wensel, A., & Huala, E. (2012). The Arabidopsis Information Resource (TAIR): Improved gene

annotation and new tools. *Nucleic Acids Research*, *40*(D1), D1202–D1210. https://doi.org/10.1093/nar/gkr1090

Pertea, G., & Pertea, M. (2020). GFF Utilities: GffRead and GffCompare. *F1000Research*, *9*, 304. https://doi.org/10.12688/f1000research.23297.1

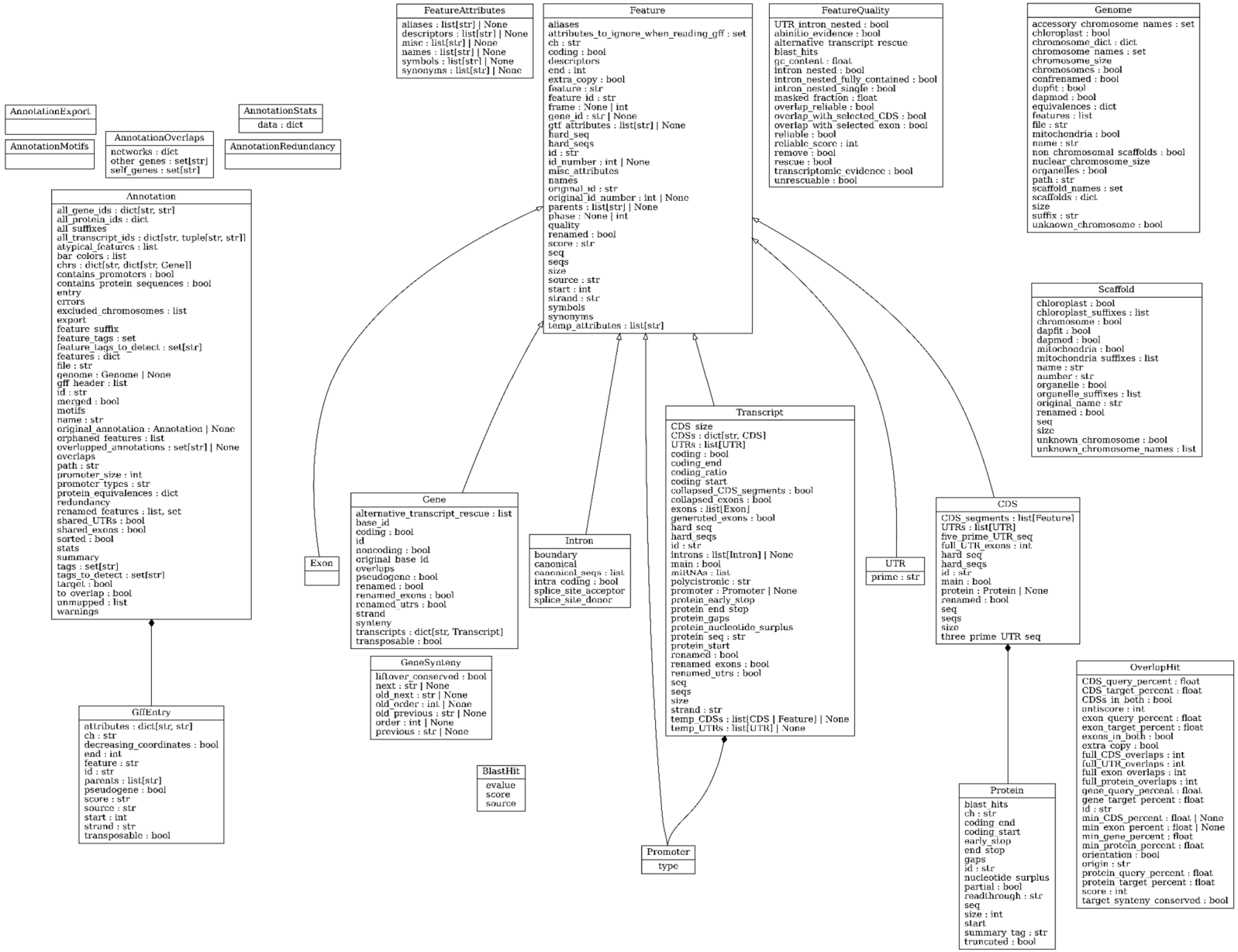
Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., & Mesirov, J. P. (2011). Integrative genomics viewer. *Nature Biotechnology*, *29*(1), 24–26. https://doi.org/10.1038/nbt.1754

Shi, X., Cao, S., Wang, X., Huang, S., Wang, Y., Liu, Z., Liu, W., Leng, X., Peng, Y., Wang, N., Wang, Y., Ma, Z., Xu, X., Zhang, F., Xue, H., Zhong, H., Wang, Y., Zhang, K., Velt, A., … Zhou, Y. (2023). The complete reference genome for grapevine (*Vitis vinifera* L.) genetics and breeding. *Horticulture Research*, *10*(5), Article 5. https://doi.org/10.1093/hr/uhad061

Shumate, A., & Salzberg, S. L. (2021). Liftoff: Accurate mapping of gene annotations. *Bioinformatics*, *37*(12), 1639–1643. https://doi.org/10.1093/bioinformatics/btaa1016

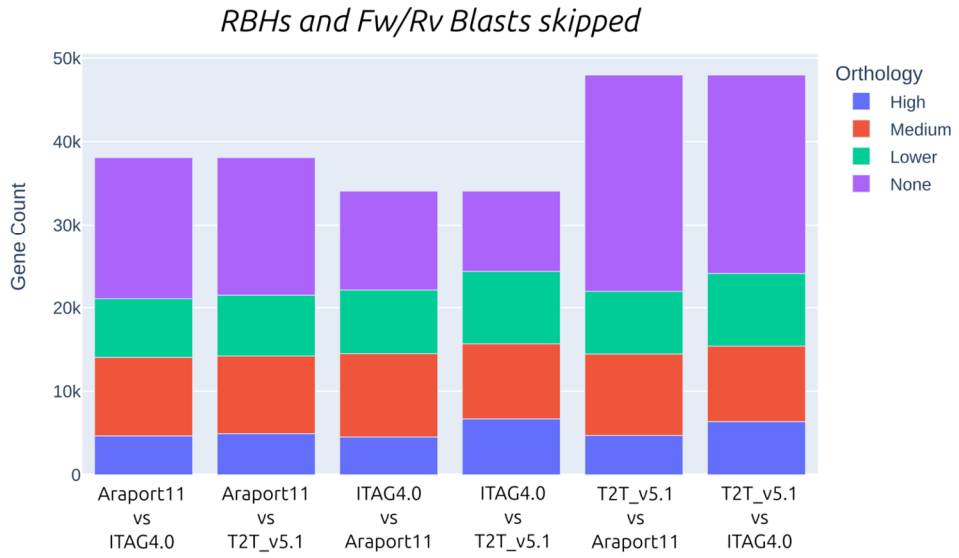
Tang, H., Krishnakumar, V., Zeng, X., Xu, Z., Taranto, A., Lomas, J. S., Zhang, Y., Huang, Y., Wang, Y., Yim, W. C., Zhang, J., & Zhang, X. (2024). JCVI: A versatile toolkit for comparative genomics analysis. *iMeta*, *3*(4), e211. https://doi.org/10.1002/imt2.211

Supplementary Information


AnnotationExport
AnnotationMotifs
AnnotationOverlaps
networks : dict
other_genes : set[str]
self_genes : set[str]
AnnotationStats
data : dict
AnnotationRedundancy
Annotation
all_gene_ids : dict[str, str]
all_protein_ids : dict
all_suffixes
all_transcript_ids : dict[str, tuple[str, str]]
atypical_features : list
bar_colors : list
chrs : dict[str, dict[str, Gene]]
contains_promoters : bool
contains_protein_sequences : bool
entry
errors
excluded_chromosomes : list
export
feature_suffix
feature_tags : set
feature_tags_to_detect : set[str]
features : dict
file : str
genome : Genome | None
gff_header : list
id : str
merged : bool
motifs
name : str
original_annotation : Annotation | None
orphaned_features : list
overlapped_annotations : set[str] | None
overlaps
path : str
promoter_size : int
promoter_types : str
protein_equivalences : dict
redundancy
renamed_features : list, set
shared_UTRs : bool
shared_exons : bool
sorted : bool
stats
summary
tags : set[str]
tags_to_detect : set[str]
target : bool
to_overlap : bool
unmapped : list
warnings
GffEntry
attributes : dict[str, str]
ch : str
decreasing_coordinates : bool
end : int
feature : str
id : str
parents : list[str]
pseudogene : bool
score : str
source : str
start : int
strand : str
transposable : bool
FeatureAttributes
aliases : list[str] | None
descriptors : list[str] | None
misc : list[str] | None
names : list[str] | None
symbols : list[str] | None
synonyms : list[str] | None
Feature
aliases
attributes_to_ignore_when_reading_gff : set
ch : str
coding : bool
descriptors
end : int
extra_copy : bool
feature : str
feature_id : str
frame : None | int
gene_id : str | None
gtf_attributes : list[str] | None
hard_seq
hard_seqs
id : str
id_number : int | None
misc_attributes
names
original_id : str
original_id_number : int | None
parents : list[str] | None
phase : None | int
quality
renamed : bool
score : str
seq
seqs
size
source : str
start : int
strand : str
symbols
synonyms
temp_attributes : list[str]
FeatureQuality
UTR_intron_nested : bool
abinitio_evidence : bool
alternative_transcript_rescue
blast_hits
gc_content : float
intron_nested : bool
intron_nested_fully_contained : bool
intron_nested_single : bool
masked_fraction : float
overlap_reliable : bool
overlap_with_selected_CDS : bool
overlap_with_selected_exon : bool
reliable : bool
reliable_score : int
remove : bool
rescue : bool
transcriptomic_evidence : bool
unrescuable : bool
Genome
accessory_chromosome_names : set
chloroplast : bool
chromosome_dict : dict
chromosome_names : set
chromosome_size
chromosomes : bool
confrenamed : bool
dapfit : bool
dapmod : bool
equivalences : dict
features : list
file : str
mitochondria : bool
name : str
non_chromosomal_scaffolds : bool
nuclear_chromosome_size
organelles : bool
path : str
scaffold_names : set
scaffolds : dict
size
suffix : str
unknown_chromosome : bool
Scaffold
chloroplast : bool
chloroplast_suffixes : list
chromosome : bool
dapfit : bool
dapmod : bool
mitochondria : bool
mitochondria_suffixes : list
name : str
number : str
organelle : bool
organelle_suffixes : list
original_name : str
renamed : bool
seq
size
unknown_chromosome : bool
unknown_chromosome_names : list
Exon
Gene
alternative_transcript_rescue : list
base_id
coding : bool
id
noncoding : bool
original_base_id
overlaps
pseudogene : bool
renamed : bool
renamed_exons : bool
renamed_utrs : bool
strand
synteny
transcripts : dict[str, Transcript]
transposable : bool
GeneSynteny
liftover_conserved : bool
next : str | None
old_next : str | None
old_order : int | None
old_previous : str | None
order : int | None
previous : str | None
Intron
boundary
canonical
canonical_seqs : list
intra_coding : bool
splice_site_acceptor
splice_site_donor
BlastHit
evalue
score
source
Transcript
CDS_size
CDSs : dict[str, CDS]
UTRs : list[UTR]
coding : bool
coding_end
coding_ratio
coding_start
collapsed_CDS_segments : bool
collapsed_exons : bool
exons : list[Exon]
generated_exons : bool
hard_seq
hard_seqs
id : str
introns : list[Intron] | None
main : bool
miRNAs : list
polycistronic : str
promoter : Promoter | None
protein_early_stop
protein_end_stop
protein_gaps
protein_nucleotide_surplus
protein_seq : str
protein_start
renamed : bool
renamed_exons : bool
renamed_utrs : bool
seq
seqs
size
strand : str
temp_CDSs : list[CDS | Feature] | None
temp_UTRs : list[UTR] | None
Promoter
type
UTR
prime : str
CDS
CDS_segments : list[Feature]
UTRs : list[UTR]
five_prime_UTR_seq
full_UTR_exons : int
hard_seq
hard_seqs
id : str
main : bool
protein : Protein | None
renamed : bool
seq
seqs
size
three_prime_UTR_seq
Protein
blast_hits
ch : str
coding_end
coding_start
early_stop
end_stop
gaps
id : str
nucleotide_surplus
partial : bool
readthrough : str
seq
size : int
start
summary_tag : str
truncated : bool
OverlapHit
CDS_query_percent : float
CDS_target_percent : float
CDSs_in_both : bool
antiscore : int
exon_query_percent : float
exon_target_percent : float
exons_in_both : bool
extra_copy : bool
full_CDS_overlaps : int
full_UTR_overlaps : int
full_exon_overlaps : int
full_protein_overlaps : int
gene_query_percent : float
gene_target_percent : float
id : str
min_CDS_percent : float | None
min_exon_percent : float | None
min_gene_percent : float
min_protein_percent : float
orientation : bool
origin : str
protein_query_percent : float
protein_target_percent : float
score : int
target_synteny_conserved : bool

Supplementary Figure S1: UML diagram of AEGIS Python classes and their attributes. Custom classes have been made for each annotation feature and wherever useful inheritance has been used to ensure code efficiency and reliability. All of the classes have been exposed at the top level of the package so they can be directly imported in Python, i.e. '*from aegis import Annotation*' without having to know the specific module where the class is located. A detailed UML diagram including class methods can be found in the package's main page (https://github.com/Tomsbiolab/aegis)

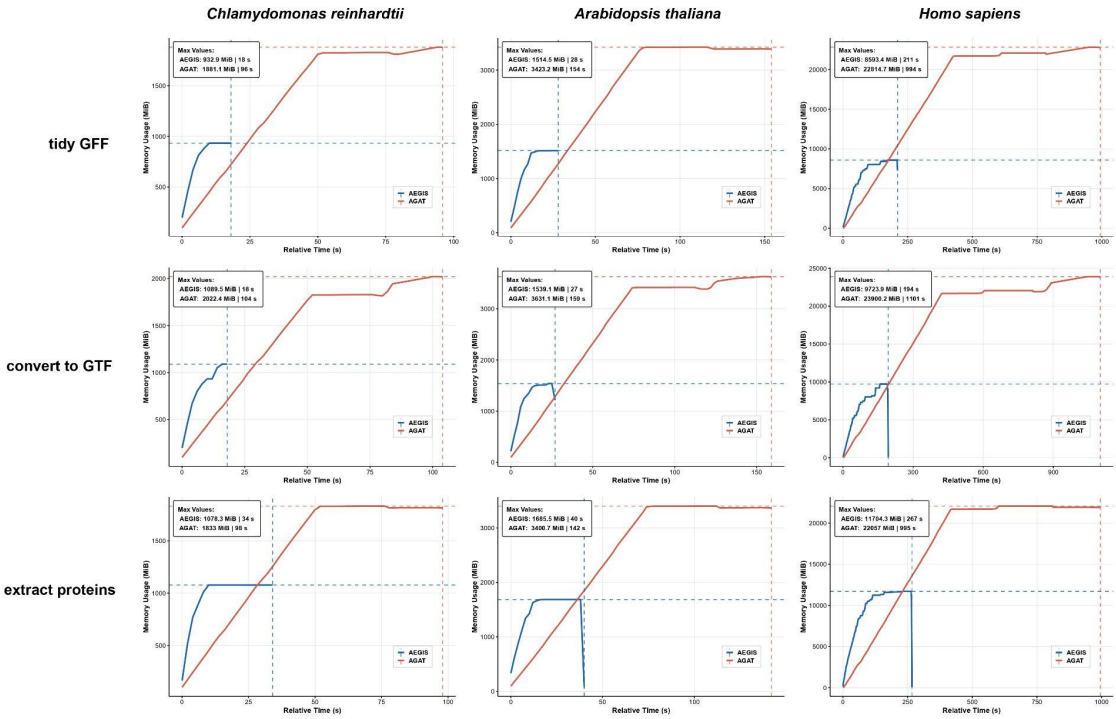
A



B

Comparison	MCscan (last filtered)	MCscan (anchors)	RBBH	Litfoff (aegis-overlaps)	LitfOn (aegis-overlaps)	OrthoFinder	None	*Aegis-orthology* (all tools combined except RBH and single blasts)	TOTAL Genes in Annotation
Araport11_ITAG4_0	18,474	11,242	12,324	1,044	4,912	17,172	17,013	21,077	38,090
Araport11_T2T_v5_1	18,881	12,781	12,688	1,303	5,399	17,447	16,586	21,504	38,090
ITAG4_0_Araport11	18,060	10,680	12,324	1,127	4,776	17,873	11,957	22,118	34,075
ITAG4_0_T2T_v5_1	19,751	13,537	13,371	3,411	9,205	19,707	9,725	24,350	34,075
T2T_v5_1_Araport11	16,518	10,569	12,688	1,306	5,171	18,122	26,022	21,949	47,971
T2T_v5_1_ITAG4_0	17,824	12,024	13,371	3,309	8,794	20,316	23,872	24,099	47,971

Supplementary Figure S2. Results of *orthology* tool skipping secondary reciprocal and single forward or reverse BLAST hits. The ***orthology*** tool can be run with the '*--skip_rbhs*' option to obtain a smaller output table that nevertheless still manages to detect orthologues for many genes. The exact output of this ***orthology*** is included in Supplementary Table S4. **(A)** The number of genes with no detected orthologues is greater in this case than with the default options (Figure 5A). **(B)** The combined results of ***orthology*** report more genes with at least one orthologue when compared to individual tool results.



Supplementary Figure S3. Computational performance benchmark of AEGIS and AGAT. Memory usage (MB) and relative execution time (s) were compared across three representative genomes: *Chlamydomonas reinhardtii* (small), *Arabidopsis thaliana* (medium), and *Homo sapiens* (large). Performance was evaluated for three core tasks: (top) GFF tidying (AEGIS ***tidy*** vs. AGAT agat_convert_sp_gxf2gxf.pl), (middle) GTF conversion (AEGIS ***reformat*** vs. AGAT agat_convert_sp_gff2gtf.pl), and (bottom) protein extraction (AEGIS ***extract*** vs. AGAT agat_sp_extract_sequences.pl).