PyamilySeq: Exposing the fragility of conventional gene (re)clustering and pangenomic inference methods

Nicholas J Dimonaco [1,2] *
[1]Institute for Global Food Security, School of Biological Sciences, Queen's University Belfast, Belfast, BT9 5DL, Northern Ireland, UK
[2]Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, Wales, UK

Abstract

Pangenomics, the identification of shared genes across a taxonomic range, is essential for understanding microbial genetic diversity and functional capacities.

This study introduces PyamilySeq, a flexible and transparent framework designed to systematically identify challenges in gene clustering and pangenomic analysis, and to support the development of practical solutions. By evaluating widely used gene clustering and pangenome tools, we can observe how clustering thresholds (often hard-coded or provided without a clear reason) and paralog handling impact gene family composition. More strikingly, while such tools are operated under the assumption that running them with broadly the same parameters will yield consistent results, this study demonstrates how parameters unrelated to clustering thresholds, such as parameter decimal precision (0.8 vs. 0.80), output selection, and even CPU and memory allocation, can alter gene family assignments. Additionally, sequence clustering and pangenome tools often fail to report biologically meaningful or representative sequences for gene families, further complicating downstream analyses.

This work highlights key limitations in current gene clustering and pangenome methodologies, demonstrating their potential to influence biological interpretations in fundamental ways. To advance the field, we must prioritise adaptable and transparent approaches that refine gene clustering methodologies and move beyond rigid, one-size-fits-all tools and parameter choices.

*To whom correspondence should be addressed. nicholas@dimonaco.co.uk

1 Introduction

Gene clustering, the process of grouping genes by shared characteristics such as function, structure, or, most often, sequence, is essential for understanding microbial diversity and functional potential. As genome databases expand in both size and complexity, traditional deterministic approaches to sequence comparison are increasingly replaced by heuristic algorithms to improve computational efficiency. Yet these heuristics introduce approximations that can compromise clustering accuracy [1]. Nevertheless, one key approach to derive biological significance from gene clusters is pangenome inference. A pangenome represents the full complement of genes within a species' range, encompassing both core genes shared by all, or most (depending on definition), strains and accessory genes that contribute to genetic diversity and adaptation [2]. Pangenomics plays a crucial role in fields such as antimicrobial resistance surveillance, pathogen detection, and evolutionary analysis by identifying core and accessory genes that influence adaptation, virulence, and resistance across a species' range[3]. However, the precision of these insights depends on reliable gene clustering, as errors in defining gene families can distort downstream biological interpretations [4, 5]. Contemporary pangenome tools are complex software systems that integrate sequence similarity and clustering methods such as BLAST [6] and CD-HIT [7]. Despite their widespread use, both sequence clustering techniques and the pangenome tools that depend on them **often lack transparency and flexibility**, raising concerns about their accuracy and reliability.

Pangenome tools typically apply additional post-clustering algorithms and grouping thresholds to refine clustering outcomes, yet the rationale behind critical steps, such as sequencing er-

ror handling, horizontal gene transfer detection, or paralog differentiation, is often poorly documented or difficult to interpret. This lack of clarity complicates efforts to determine whether observed gene clusters, or families, reflect true biological relationships or are artifacts of the tool's methodology. While some studies have assessed the broad impact of biases introduced by gene clustering and pangenome inference methods [8, 9, 10], a systematic and detailed evaluation of their effects is still lacking.

Adding to these issues, genome assembly and annotation remain major sources of variability, often introducing inconsistencies that directly influence clustering accuracy [11, 12]. Completeness of gene function classification is a significant problem still to be resolved for even the most well studied genomes, such as *E. coli* which has only around 60% of it's genes "Well-characterized" as recently as 2024 [13]. However, their impact on clustering results remains poorly understood. Although some tools attempt to correct annotation inaccuracies before clustering, they are still reliant on previous knowledge, alignment to known genes, or require additional experimental evidence such as RNA and proteomic analysis [14, 15]. The extent to which annotation errors and clustering biases collectively distort pangenome inference is an open question that must be explored in greater depth.

A variety of pangenome tools exist, each employing different methodologies. In this study, I focus on two: Roary [16], one of the most widely used tools with nearly five thousand citations, and Panaroo [15], which represents a more recent advancement in computational pangenome analysis.

Roary [16], once a cornerstone of the field, has not received updates since 2019 (`github.com/sanger-pathogens/Roary`). The project's GitHub repository currently states: *"PLEASE NOTE: we currently do not have the resources to provide support for Roary, so please do not expect a reply if you flag any issue"*. Consequently, the repository now primarily serves as a platform for community discussion, where users have shared unresolved issues, feature suggestions, and bug reports. Despite this, Roary remains widely used and is still incorporated into contemporary workflows [17]. Unfortunately, this situation is common in bioinformatics, where essential tools become unsupported over time due to the lack of funding priority and resources. In contrast, Panaroo [15] continues to be actively developed and has introduced assembly and gene prediction error correction and improved gene relationship modeling.

A key factor influencing the accuracy of both Roary and Panaroo is their reliance on external clustering algorithms, particularly CD-HIT [7]. Designed for high-throughput sequence similarity clustering, CD-HIT plays a fundamental role in many bioinformatic workflows. However, its reliance on often misunderstood parameters, one-size-fits-all thresholds, and heuristic methods means that small parameter and input changes can drastically alter clustering outcomes. These rigid parameters, often set arbitrarily or based on historical defaults, may force unrelated genes into the same group or split genuinely related genes apart, leading to biased representations of genomic relationships. Understanding how these biases manifest across different tools and parameter settings is crucial for improving pangenome inference.

Given the complex interplay between annotation quality, sequence clustering, and pangenome inference, there is a pressing need for tools that can systematically examine how methodological choices influence gene clustering and, therefore, gene family outcomes. To address this, PyamilySeq was developed as a diagnostic framework for studying both gene clustering and pangenome inference methodologies. However, throughout its development, PyamilySeq evolved into a versatile and fully-featured pangenome analysis tool. It not only replicates the core functionalities of widely used pangenome tools such as Roary and Panaroo but also their outputs including gene family designations as follows: Core genes ($99\% \geq$ strains $= 100\%$), Soft core genes ($95\% \geq$ strains $< 99\%$), Shell genes ($15\% \geq$ strains $< 95\%$) and Cloud genes ($0\% \geq$ strains $< 15\%$). PyamilySeq also provides increased transparency in clustering decisions, flexibility in parameterisation, and iterative reclustering capabilities. In doing so, PyamilySeq serves both as a standalone pangenomic toolkit and an investigative engine for probing the assumptions and biases that underpin contemporary gene clustering methodologies. By varying key parameters and analysing their impact on gene family composition, I reveal how clustering thresholds, algorithmic decisions, sequence alignment, paralog handling and even runtime factors such as CPU and memory allocation shape pangenomic outputs. While PyamilySeq was developed with processes that may be broadly useful beyond this study, my objective is not to position it as a re-

placement for existing tools. Rather, this work aims to critically evaluate current methodologies, offering deeper insight into the factors shaping clustering results and guiding the development of more transparent, flexible, and biologically meaningful pangenome analysis approaches.

2 Results

The results presented in this section were generated using a new tool developed specifically for this study, PyamilySeq. While a detailed description is provided in the Methods, a brief overview is given here to orient the reader.

PyamilySeq is a pangenome analysis toolkit designed to investigate each stage of gene clustering and pangenome inference. It begins by parsing the output of clustering tools such as CD-HIT, then traces the origin of each gene within clusters to specific genomes, and finally determines the distribution of each gene family e.g., core, soft-core, or accessory, across the dataset. PyamilySeq is particularly suited for identifying subtle differences introduced by small parameter changes or input variations, offering insight into how such factors may influence results in sequence clustering or other pangenome tools. It also includes a suite of auxiliary utilities for interrogating intermediate files and interpreting both inputs and outputs across runs.

PyamilySeq is applied to two datasets: one comprising 10 *E. coli* genomes with varying levels of assembly fragmentation, used to examine how assembly quality may influence pangenome inference; and a second dataset of 74 *E. coli* genomes, each assembled into fewer than five contigs, providing a high-quality, 'well-characterised' genome set for robust comparative analysis. Lastly, PyamilySeq is agnostic to sequence type, allowing analysis of gene families using both nucleotide (**DNA**) and amino acid (**AA**) sequences. In this study, we can observe this flexibility to compare gene family composition across both sequence types for each genome set. This dual approach enables us to explore and account for limitations and differences inherent to clustering and alignment tools operating in DNA versus protein space. Throughout this study, **PyamilySeq DNA** and **PyamilySeq AA** will be used to explicitly state which sequence type analysis is being performed or discussed. An overview of how PyamilySeq works is presented in Figure 1.

To account for different definitions of core, soft

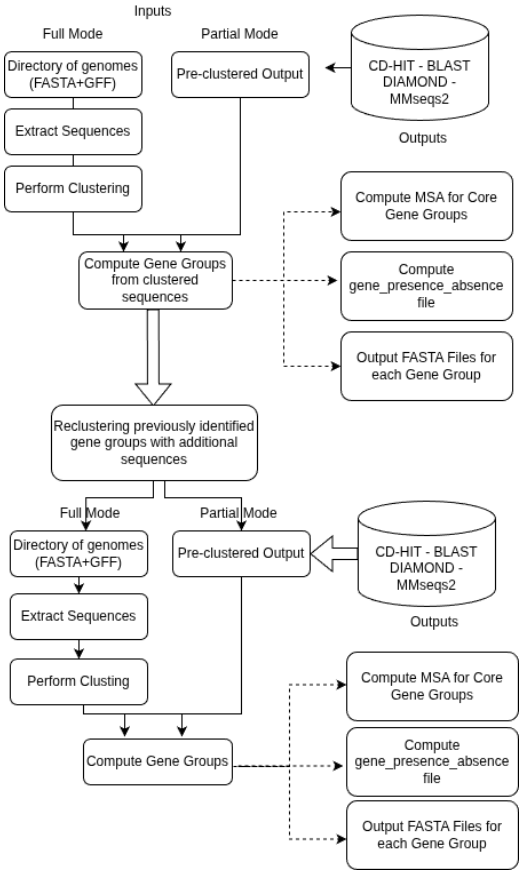

Figure 1: Overview of the PyamilySeq toolkit workflow. PyamilySeq offers two operational modes: Full Mode, which processes genome annotations (GFFs) to extract sequences and perform *de novo clustering* with CD-HIT, and Partial Mode, which allows users to input pre-clustered gene groups for further analysis. Clustering is performed using CD-HIT in Full mode or BLAST, DIAMOND and MMseqs2 in Partial mode, generating gene families that can be further refined through reclustering. The toolkit outputs key files including multiple sequence alignments for core gene groups, gene presence-absence matrices, and FASTA files for each gene group, providing a comprehensive framework for pangenome analysis.

core and accessory gene families, PyamilySeq has been designed to transparently report the percentage ranges used for easier comparison. Additionally, to account for PyamilySeq's ability to track existing and additional sequences during its reclustering step, additional metrics were required. Below is a detailed explanation of the reporting metrics used by PyamilySeq which when used with default parameters, match the core, soft core and accessory definitions of both Rorary and Panaroo.

- **First_core_#**: (First Core) Gene groups identified in the **initial round of clustering** before any reclustering is performed - These results are directly comparable to Roary and Panaroo.
- **extended_core_#**: Gene groups from the initial clustering that were **expanded** by incorporating sequences from the reclustered dataset during the second round of clustering.
- **combined_core_#**: Gene groups where sequences from the reclustered dataset **merged two or more original clusters** into a single, larger group (see Figure 11).
- **Second_core_#**: (Second Core) New gene groups that are classified as core **exclusively from the reclustered sequences**, without contributions from the initial clustering - First round sequences may be present but are not counted.
- **only_Second_core_#**: A subset of **Second_core_#** containing gene groups that consist **entirely of reclustered sequences**, with no overlap with the initial dataset.

List 1: Summary of the gene group classifications in PyamilySeq output. The notation follows the structure used in the StORF-Reporter publication [18], distinguishing between **First Core (FC)** - results from the initial clustering, and **Second Core (SC)** - results from reclustering. The # symbol denotes the percentage identity thresholds used for defining gene groups, with default values set to 99%, 95%, 15%, and 0%.

Both datasets and their preparation are described in detail in the Methods section.

2.1 Pangenomic analysis of 10 *Escherichia coli* genomes reports a consensus of approaches

First, PyamilySeq was applied to a dataset of 10 *Escherichia coli* genomes with sequence identity set to 0.90 and a `length difference cutoff` of 0.80. The results are summarised below (two sequences in amino acid form were excluded from the PyamilySeq AA analysis as CD-HIT cannot handle in-sequence stop codons - '`*`').

Groups	PyamilySeq[DNA]	PyamilySeq[AA]	Roary	Panaroo
Core genes:	2989	2996	3,078	3,309
Soft core genes:	0	0	0	0
Shell genes:	3,309	3,259	3,463	3,027
Cloud genes:	5,314	5,469	6,181	2,607
Total genes:	11,612	11,724	12,722	8,943

Table 1: This table reports the gene family composition reported for each tool on the set of 10 *E. coli* genomes.

A number of key observations can be made from the comparison of the summary results generated by the three tools, PyamilySeq, Roary and Panaroo. Notably, the number of core-gene groups reported by PyamilySeq, falls just slightly below the results reported by Roary and Panaroo. This suggests that, under certain conditions and clustering parameters, PyamilySeq reports comparable core gene numbers to contemporary tools, even though the aim of this study and the development of PyamilySeq was not to directly compare it with existing methods. This initial comparison serves as a valuable 'sense check', demonstrating that PyamilySeq, when configured with relatively strict CD-HIT clustering parameters, aligns reasonably well with established tools. Such results suggests that PyamilySeq can provide a reliable basis for exploring the influence of clustering parameters on gene family assignments, helping to identify potential biases and opportunities for refinement in clustering and pangenomic methods.

The three tools each generated a DNA core gene (defined as present in 99% of genomes) multiple sequence alignment (MSA), that were subsequently used to build phylogenetic trees. The tools independently align each gene family using MAFFT [19], and then concatenate the alignments to produce a complete MSA for tree construction. Figure 2 shows the DNA-based phylogenetic trees generated using FastTree2 [20] from the MSAs produced by Roary, Panaroo, and PyamilySeq, respectively. As illustrated, there are little discernible structural differences between the trees produced by the three tools, demonstrating that

they yield consistent results when analysing this relatively small and homogeneous dataset.

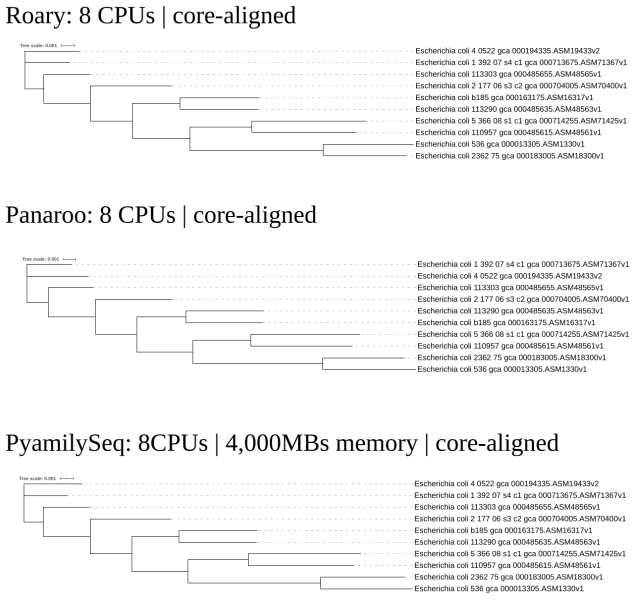

Figure 2: Three phylogenetic trees built using the 'core-gene' families identified by Roary, Panaroo and PyamilySeq.

To further confirm the similarity of the phylogenetic trees generated by each tool, a Robinson-Foulds (RB) [21] symmetric difference test was performed using the Python package ETE 3 [22]. RB can be used to measure the dissimilarity between two phylogenetic trees by quantifying the number of shared splits and normalising it to a maximum possible distance, typically ranging from 0 to 1, where 1 is most different. This test revealed that while there were minor rearrangements for some of the branches, the RB analysis reported no differences between the trees, reinforcing that all three tools produce equivalent phylogenetic results for this dataset.

2.2 Pangenomic analysis of 74 *Escherichia coli* genomes reports significant differences based on approach

Although the previous analysis of 10 *E. coli* genomes showed minimal differences between clustering approaches, most pangenomic studies involve a larger and more diverse set of genomes. Expanding the analysis to 74 genomes reveals substantial discrepancies in core and accessory gene assignments, highlighting the impact of clustering methodology on pangenome inference. The results are summarised in Table 2 below.

As the number of *Escherichia coli* genomes analysed increases, discrepancies between the three tools, Roary, Panaroo, and PyamilySeq, become more pronounced. Unlike the 10-genome study, where core gene counts were relatively similar, each tool now reports a substantially different number of core genes.

Groups	PyamilySeq[DNA]	PyamilySeq[AA]	Roary	Panaroo
Core genes:	432	357	1,023	2,824
Soft core genes:	2,121	2,139	860	472
Shell genes:	3,185	3,220	4,696	3,040
Cloud genes:	34,908	37,014	31,890	15,190
Total genes:	40,646	42,730	38,469	21,526

Table 2: This table reports the gene family composition reported for each tool on the set of 74 *E. coli* genomes.

Using DNA-based clustering, PyamilySeq identifies 432 core gene families, compared to 357 using its AA-based clustering, both significantly lower than the 1,023 core genes reported by Roary. Panaroo, in contrast, reports an entirely different distribution, identifying nearly three times as many core genes as Roary and almost seven times more than PyamilySeq. These discrepancies raise some fundamental questions: Which tool provides the most accurate representation of the pangenome? Or, do these differences instead reflect the inherent complexity and subjectivity of pangenome analysis itself?

Somewhat surprisingly, despite these substantial differences in core gene counts, the phylogenetic trees built from each tool's core gene families (Figure 3) appear broadly similar. Robinson-Foulds analysis further supports this, reporting relatively small differences between trees. Each tool comparison: Roary vs. PyamilySeq, Roary vs. Panaroo, and Panaroo vs. PyamilySeq, yielded an identical Robinson-Foulds distance of 24 out of a possible 142, normalised to 0.169.

Although this is a single-species comparison, it may explain why discrepancies in core gene family counts, like those observed here, are not always widely reported. If one of the primary reasons for undertaking a pangenome analysis is phylogenetic inference, and these discrepancies do not drastically alter tree topology, they may go unnoticed. However, this does not mean such differences are scientifically acceptable. While phylogenetic trees may be minimally affected, studies on functional conservation across species could be heavily impacted by the variation in core gene family definitions. Figure 4 and Supplementary Table 5 present broad functional classifications of the core gene families identified by each tool, assigned using the eggNOG-mapper tool [23]. While we may

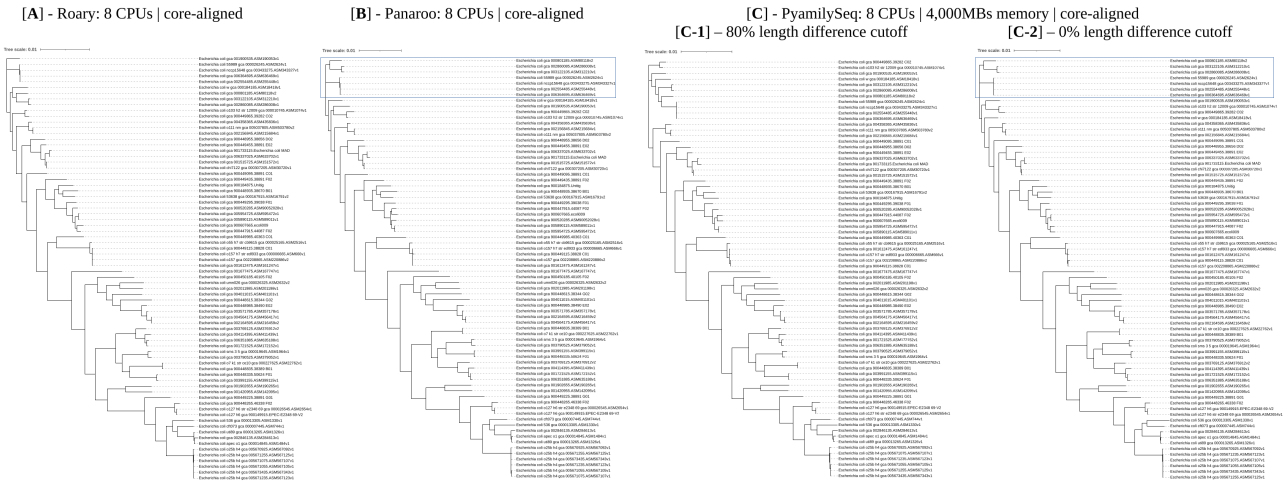

Figure 3: Three phylogenetic trees built using the 'core-gene' families identified by Roary (A), Panaroo (B) and PyamilySeq[AA] (C) for the 74 genomes.

expect an increase in the proportion of genes being reported as 'POORLY CHARACTERIZED' as the core increases, somewhat unexpectedly, we instead observe a 6% increase in genes assigned to 'METABOLISM' and instead a reduction in 'POORLY CHARACTERIZED' genes. While not conclusive, this result suggests a possible trend: by inflating core gene family counts, we may not only be losing functional resolution but also altering the specificity of the core gene functional landscape.

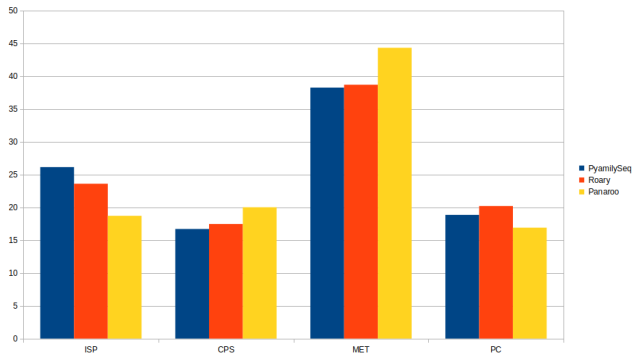

Figure 4: Broad functional overview from the core gene sets reported by PyamilySeq, Roary and Panaroo using COG categories from eggNOG-mapper.

2.3 Influence of CPU, memory, parameter precision (e.g., -x 0.8 vs 0.80) and core-gene alignment on gene clustering results

During PyamilySeq's development and testing, small discrepancies were observed in gene grouping counts between Full and Partial modes, despite using identical CD-HIT clustering parameters. While it is well known that heuristic-based sequence alignment tools such as CD-HIT and BLAST can yield different results based on randomised seeds, the impact of system resource allocation, specifically in CD-HIT, CPU (Central Processing Unit - Users can set the number of CPU threads using '-T' to improve performance) and memory provisioning (selectable with '-M', is the amount of memory in Megabytes (MB) CD-HIT can use during runtime to also improve performance) on clustering results is less widely recognised. As seen in Supplementary Table 3, passing varying CPU and memory allocations to CD-HIT when running PyamilySeq results in small but observable differences in the core gene assignments. These results suggest that computational resource allocation can influence clustering outcomes beyond just performance.

While the number of First Core (FC) genes remains almost unchanged across different CD-HIT parameter runs, some genes were reassigned to different clusters, indicating a rearrangement in their grouping. For example, in PyamilySeq's default CD-HIT run (-T 8 -M 4000), the singleton gene 'ENSB_nhmgO-nwCXecCuN' remained unclustered. However, under CD-HIT's default settings (-T 1 -M 800), it clustered with another sequence, 'ENSB_EF9iv1mvUVBoPlR' (PID: 90.16%), from a different genome. These results are consistent across multiple interactions with the same parameters, thus further indicating that they are a direct result of parameter changes and not just run-to-run heuristics. Notably, in the PyamilySeq (-

T 8 -M 4000) run, 'ENSB_EF9iv1mvUVBoPlR' was grouped with 25 other sequences across 22 genomes. To determine whether clustering discrepancies increase with higher CPU and memory settings, CD-HIT was executed with even higher values (-T 12 -M 40000). Surprisingly, 'ENSB_nhmgO-nwCXecCuN' clustered with 'ENSB_EF9iv1mvUVBoPlR', mirroring the default (-T 1 -M 800) results rather than the PyamilySeq '**-T 8 -M 4000**' parameter run. While this is an issue specifically attributable to the CD-HIT clustering and not PyamilySeq (or Roary/Panaroo), it must be accounted for by at minimum reporting the performance parameters with the results of the analysis. Supplementary Table 4 reports the CD-HIT initialisation outputs for the aforementioned different `-T (CPU)` and `-M (Memory)` parameter choices. What we can see in Supplementary Table 4, is CD-HIT utilising different numbers of 'representatives' and 'word counting entries' depending on the selection of CPU and Memory parameters, effectively creating unintentional seed changes (see Supplementary Listings 1-3 for a more detailed view).

Given that CD-HIT serves as the primary clustering step in both Roary and Panaroo, I tested whether varying CPU parameters affected their gene family assignments on the set of 74 *E. coli* genomes. As shown in Table 5, even with a relatively small dataset of 74 genomes, varying CPU allocations (2, 4, and 8 threads) resulted in differences in reported gene families for both Roary and Panaroo. As a consequence of this, all analysis reported for PyamilySeq, Roary and Panaroo use the same CPU parameters of 8 threads and additionally all PyamilySeq analyses was conducted with memory allocation set to 4,000MB (-M 4000) for its internal CD-HIT run.

Another unexpected factor influencing CD-HIT clustering results was the precision of the -c (`sequence identity threshold`) and -s (`length difference cutoff`) parameters. Specifically, using one vs. two decimal places in parameters selection (e.g., 0.9 vs. 0.90) led to observable differences in clustering outcomes. In the 74-genome *E. coli* dataset, these small variations resulted in some accessory genes being grouped differently. This effect possibly stems from differences in how programming languages handle floating-point precision and rounding behaviour. While detailing further as to why this happens is not within the scope of this study, the results are consistent across runs on both x86 Linux and ARM Apple Silicon systems.

These findings suggest that the interaction between CPU allocation, memory, and parameter precision introduces a degree of non-linearity in clustering outputs. Complicating matters further, it remains unclear how Roary and Panaroo internally handle and pass user-defined parameters, making it difficult to fully control for these sources of variability.

Perhaps the most unexpected clustering inconsistency arises when running Roary in alignment mode. As seen in Table 4, enabling core gene alignment (producing a concatenated multiple sequence alignment across all genomes with each reported core gene) with either MAFFT or PRANK [24] influences the distribution of core and accessory genes, an effect that becomes more pronounced in larger datasets.

2.4 Does it matter whether we cluster on DNA or AA?

Pangenomic inference is typically performed on amino acid (AA) sequences, as functional conservation is believed to reside primarily in the protein sequence. While significant advancements have been made in annotating protein-coding regions of DNA [12], determining the exact amino acids encoded by a given codon remains a complex challenge, with current computational methods fundamentally lacking [25].

While the total number of clusters using PyamilySeq remained nearly identical in the 10-genome study (2,989 core gene families in DNA vs. 2,996 in AA), a larger discrepancy emerged in the 74-genome study, where core gene families decreased from 432 (DNA) to 357 (AA). These small-scale examples illustrate key differences to both DNA and AA sequence clustering. To explore this further, I examined the largest cluster identified in the 74-genome study. In the DNA clustering, CD-HIT(-EST) formed a 506 gene cluster, selecting *ENSB_s-GX_UiVmBsWL9K* (291 nt) as its representative sequence. The most divergent sequence within this cluster, *ENSB_Vyjslictzez3CNX* (249 nt), had a reported 98.39% sequence identity to the representative. These two sequences effectively define the range of conservation or difference within this cluster. They are the centroid and the most distinct/different sequence. However, in the AA sequence clustering, both sequences were assigned as singletons, despite clustering being performed under identical default parameters. This discrepancy stems from an inherent limita-

Groups	Roary 2	Roary 4	Roary 8	Panaroo 2	Panaroo 4	Panaroo 8
Core genes	1,030	1,031	1,023	2,824	2,824	2,824
Soft core genes	859	860	872	472	472	472
Shell genes	4,743	4,720	4,740	3,027	3,033	3,040
Cloud genes	31,759	31,824	31,709	15,195	15,163	15,189
Total genes	38,391	38,435	38,344	21,518	21,492	21,525

Table 3: This table reports the different gene family results reported by Roary and Panaroo on the 74 **E. coli** dataset with default parameters but different CPU allocations.

Groups	Roary 10 *E. coli* genomes			Roary 74 *E. coli* genomes		
	No-Align	MAFFT	PRANK	No-Align	MAFFT	PRANK
Core genes	3,078	3,078	3,077	1,023	1,026	1023
Soft core genes	0	0	0	872	848	866
Shell genes	3,470	3,463	3,458	4,740	4,763	4,709
Cloud genes	6,172	6,181	6,187	31,709	31,714	31,799
Total genes	12,720	12,722	12,722	38,344	38,351	38,397

Table 4: The influence that the align core genes option with PRANK (`-e`) and MAFFT `-e & -n`) has on the output of Roary while all other parameters are set to default and the CPU allocation is fixed to 8.

tion of CD-HIT's clustering algorithm: gaps do not contribute to sequence differences, meaning alignments with an arbitrary number of gaps can still report 100% identity.

Although CD-HIT does not provide an option to visualise alignments, we can potentially infer its behaviour using BLAST and Clustal-Omega [26, 27] by observing how other coding-frame unaware sequence aligners operate. As shown in Figure 5, both `blastn` and `blastp` report alignments between *ENSB_s-GX_UiVmBsWL9K* and *ENSB_Vyjslictzez3CNX* with a percentage identity above 90%, whereas Clustal-Omega does not. In particular, while a 90% identity cut-off in Clustal-Omega would exclude these sequences, their DNA alignment score of 89.3% is much higher than their AA alignment of 78.3%. This discrepancy likely explains why CD-HIT clusters them in DNA but not in AA. Additionally, the alignments reveal gaps that may be misinterpreted by alignment tools lacking coding-frame awareness, such as CD-HIT and BLAST. Notably, the only annotated protein domain information for these sequences corresponds to the aligned segments, which are classified as intrinsically disordered regions, possibly further complicating reliable alignment and clustering.

Lastly, an extreme case that further highlights the limitations of alignment tools such as BLAST. Here we have two sequences of vastly different lengths, *ENSB_U6Q8FqhU_eaDP5e* (1,119 nt) and *ENSB_8ANT4ngdQh30mf9* (2,091 nt), yield no significant similarity in `blastn` (Figure 6A), yet `blastp` aligns them with 90% identity over their shared region (Figure 6C). Panels **B** and **C** illustrate a scenario closer to the behaviour of CD-HIT, where higher AA sequence identity than DNA sequence identity leads to clustering. Although such cases are likely rare, they can be partially explained by codon optimisation and reassignment [28, 29]. Nonetheless, they underscore a broader issue that sequence alignment and clustering tools used for pangenomic inference are often ill-equipped to fully capture biological complexity.

2.5 Paralogs, orthologs and gene duplication

Pangenomic inference tools aim to classify gene families by distinguishing between orthologs, genes that arise from speciation events and typically retain similar functions, and paralogs, genes that result from gene duplication within a genome and may evolve new or redundant roles. Accurate separation of these gene types is crucial, as misclassification can obscure evolutionary relationships, affect gene family counts, and impact downstream phylogenetic and functional analyses.

Both Roary and Panaroo perform 'paralog splitting' by default. Here we will investigate how Roary handles this process by enabling the parameter `-s dont split paralogs`, which disables paralog splitting. Roary leverages conserved gene neighbourhood information to distinguish paralogs from true orthologs. As shown in Table 6, toggling this parameter has a substantial impact, nearly tripling the number of core gene families. This underscores the importance of paralog splitting and thus the impact that any potential problems with how it is conducted could have on the final output.

To investigate the substantial increase in core gene families, I compared the *gene_presence_absence.csv* files from both Roary runs, one with default settings and the other

Sequence	Pos 1	Pos 2	Pos 3	Pos 4	Pos 5	Pos 6	Pos 7	Pos 8	Pos 9	Pos 10	..	Pos 20
DNA Seq 1	ATG	GCT	TCT	ACT	CAG	CGT	ATT	CTG	GTG	GGT	..	TGA
DNA Seq 2	ATG	GCC	AGC	ACA	CAA	AGA	ATA	TTA	GTA	GGA	..	TAG
AA Seq	M	A	S	T	Q	R	I	L	V	G	..	*

Table 5: An example of where two different nucleotide sequences can encode the same amino acid sequence using the same codon table (11). DNA column codes for the same amino acid sequence, despite different nucleotide sequences.

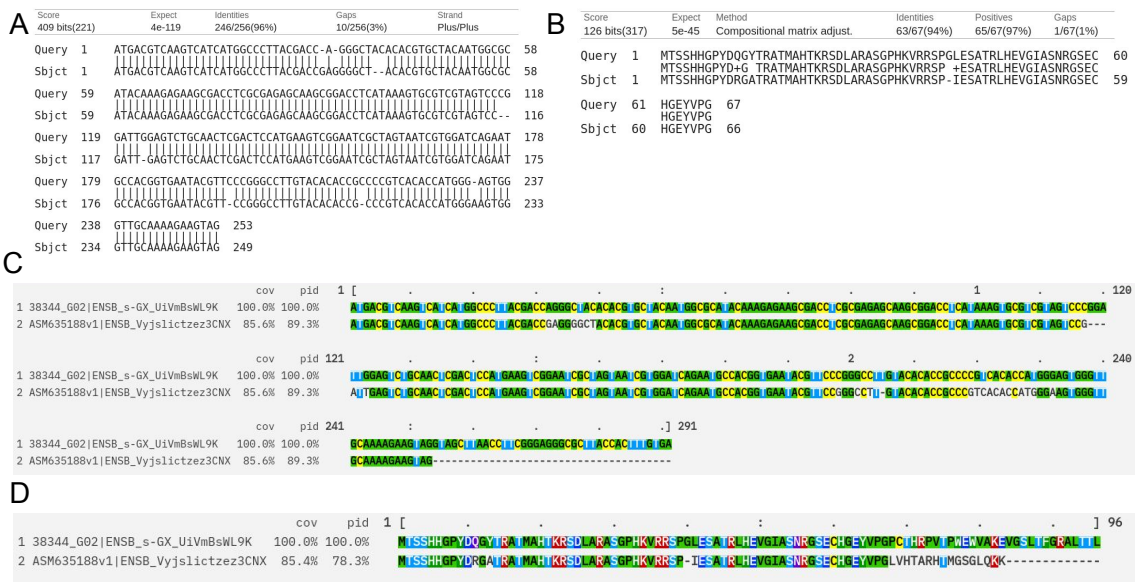

Figure 5: Panels depicting some of the problems with clustering in the DNA and AA spaces and how we get different results from different methods. Panels **A** and **B** report the BLAST alignments of sequences *ENSB_s-GX_UiVmBsWL9K* and *ENSB_Vyjslictzez3CNX* in DNA and AA, respectively, where both reported a higher than 90% percentage identity. Panels **C** and **D** report the multiple sequence alignment conducted by Clustal Omega in DNA and AA respectively but reporting a percentage identity of less than 90%.

Groups	Roary with splitting	Roary without splitting
Core genes	1,026	2,554
Soft core genes	848	602
Shell genes	4,763	2,430
Cloud genes	31,714	16,000
Total genes	38,351	21,586

Table 6: Gene family distribution of the 74 *E. coli* genomes reported by Roary with and without paralog splitting.

with `-s dont split paralogs` on the set of 74 genomes. The differences were striking. Under default parameters, Roary's largest core gene group contained 149 sequences, and 420 gene families had at least one genome with multiple gene sequences, yet only 4 of these were core gene families. However, with `-s dont split paralogs`, the largest core gene group now had 478 sequences, and of the 2,892 gene families containing at least one genome with multiple gene sequences, 2,554 were core. These findings highlight the significant impact of paralog splitting on Roary's output.

To explore how Roary assigns suspected paralogs, we examined the core gene family with the most sequences (group_21463) in the `-s dont split paralogs` output. This family contained sequences from all 74 genomes, totalling 478 genes, reporting that multiple copies existed within most genomes. Using PyamilySeq's `Seq-Finder` tool, these genes were cross-referenced in the default Roary output, revealing that they had originally been distributed across 10 separate gene families, each containing 1 to 70 genes (see Supplementary File 1). This reassignment in-part explains why Roary reports far fewer core gene families under default settings. To further investigate, the DNA and AA sequences of these 478 genes was retrieved and clustered with CD-HIT. The results showed

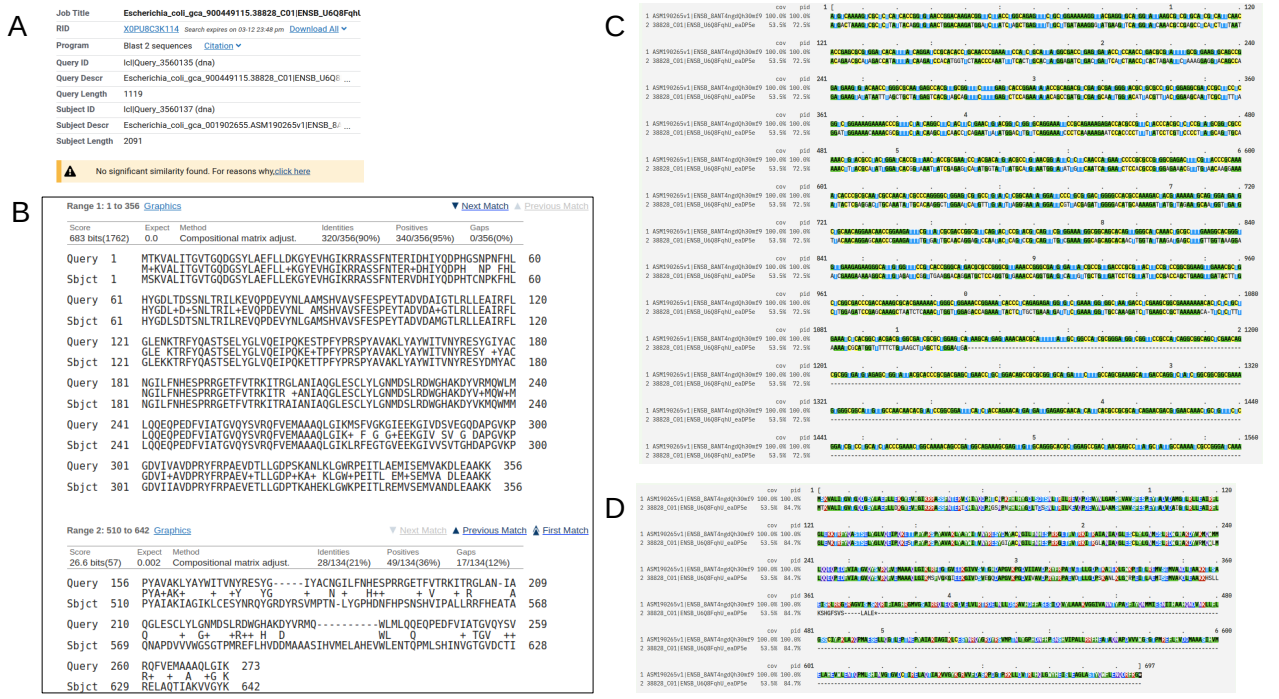

Figure 6: Panels reporting BLAST and Clustal-Omega alignment results for *ENSB_U6Q8FqhU_eaDP5e* and *ENSB_8ANT4ngdQh30mf9*. Panels **A** and **B** report how 'no significant similarity' was found by BLAST in the AA alignment, but two segments were able to be aligned in the DNA alignment, the first with 90% identity. Panels **C** and **D** shows the Clustal-Omega alignments which were reported at 72.5% and 84.7% for DNA and AA respectively.

that all 478 AA sequences were identical (84 AAs, 100% identity). At the DNA level, only 6 sequences differed, each by a single inconsistent nucleotide. Notably, these variants had other identical copies within the same genome, suggesting recent duplication events rather than true divergence. These findings indicate that Roary's default settings likely rely more on gene synteny and neighborhood context than on sequence variation alone. While this is a powerful algorithmic choice, it most-likely relies on high quality assemblies and annotation to determine each genes placement.

Further analysis of core gene families containing suspected paralogs revealed significant length variation. For example, in group_9 from the `-s dont split paralogs` output, the average gene length was 690 nucleotides, yet lengths ranged from 119 to 6,767 nucleotides. Alignment of the longest and shortest sequences in this group showed minimal similarity (Figure 7). While individual sequences were more similar to others in the group, the lack of overlap between the longest and shortest sequences raises concerns about Roary and Panaroo's default clustering criteria. Both tools leave CD-HIT's length difference cutoff at its default setting, potentially grouping sequences with extreme length variation into the same gene family. I reported this issue on Panaroo's GitHub page, and as of release 1.5.1, users can now modify CD-HIT's length difference cutoff parameter (`github.com/gtonkinhill/panaroo/releases/tag/v1.5.1`). However, the default parameter stays unchanged and requires detailed user understanding to utilise correctly.

Figure 7: Alignment of two sequences from group_9 that have low similarity of ∼40% no similarity but nonetheless were grouped together into the same gene family by Roary with the `-s dont split paralogs` options.

Although grouping genes with significant length variation may be appropriate in some cases, concealing these differences within the clustering process risks misleading conclusions. Figure 8 illustrates an example from Panaroo (default parameters) where clustering sequences of varying lengths may be beneficial. In core gene family

group_21184, the genome '40363_C01' contains a 962 AA sequence, while '39282_C02' has three shorter sequences aligning independently across different regions of the longer sequence (start, middle, and end). BLAST analysis reveals that the longer sequence is a 100% identity match to a known *E. coli* pitrilysin protein (`NCBI Reference Sequence: WP_038338917.1`), while the three shorter sequences also match the same protein, albeit with varying identities (Figure 8).

This example highlights Panaroo's ability to capture gene families despite complications from genome assembly errors, annotation inconsistencies, or gene pseudogenisation. However, the fragmentary status of these sequences is not explicitly conveyed to users, requiring expertise and manual effort to detect. Additionally, this clustering behaviour interacts with Panaroo's 'refound' system, which uses BLAST to recover potentially missing genes caused by assembly or annotation errors. Although this system has known issues, such as returning fragmented gene sequences due to pseudogenisation or assembly artefacts, it has been partially addressed in recent updates. While users now have further control including the ability to entirely disable the refound system, it is on by default and may continue to produce misleading or unexpected results (`github.com/gtonkinhill/panaroo/issues/263`). Ultimately, cases like this underscore a broader transparency challenge pangenome tools, where critical clustering decisions and potential algorithmic artefacts may not be readily apparent to users.

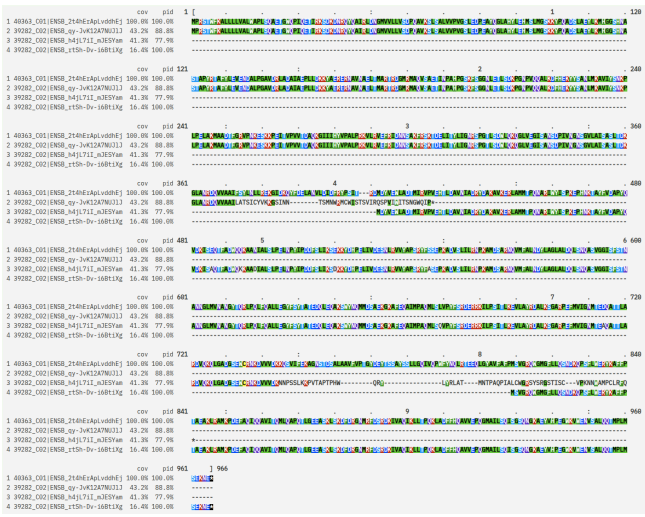

Figure 8: Alignment of sequences in core gene family group_21184, showing a 962 AA sequence and three shorter sequences aligning to different regions of the same *E. coli* pitrilysin protein.

2.6 The impact of sequence length on gene clustering

We have seen several examples that have highlighted how gene sequences of vastly different lengths were clustered together by both Roary and Panaroo. To further investigate this, PyamilySeq was rerun on the 74-genome dataset with CD-HIT's `length difference cutoff` set to 0% as it is in Roary and Panaroo by default. Although CD-HIT clusters sequences sharing >= 90% identity along aligned regions, an inherent algorithmic minimum exists for what that aligned region must be, even when set to '0%'. To assess the impact of relaxing length constraints, the Cluster-Summary tool from PyamilySeq was applied to the AA clustering results. A summary is provided in Table 7, with the full dataset available in Supplementary File 2.

Clus ID	# Seqs	# Genomes	Avg Length	Length Range	Avg Ident'	Ident' Range
0	745	61	124.51	67-232	96.23	90.42-100.00
1	502	58	199.87	29-409	97.17	90.18-100.00
9	213	33	184.5	57-449	96.87	91.23-100.00
17	153	74	510.87	88-1070	98.28	90.48-100.00
19	152	74	618.18	84-1286	99.31	91.51-100.00
20	152	74	349.03	86-736	98.69	91.53-100.00
21	150	74	439.54	64-955	98.59	90.65-100.00
55	127	63	411.06	71-799	95.57	90.28-100.00
57	125	63	430.02	77-884	94.07	90.39-100.00
63	115	59	463.9	31-1016	96.37	90.07-100.00

Table 7: Summary of 10 gene clusters with a diversity of sequence lengths but relatively conserved sequence identity ranges from the CD-HIT results of clustering AA sequences at 90% percentage identity and 0% `length difference cutoff` on the 74 *E. coli* genomes.

As can be seen in Table 8, a dramatic shift in core and soft core gene counts occurs when adjusting the `length difference cutoff` in CD-HIT. Specifically, lowering the length difference threshold allowed genes with greater size disparities to cluster together, resulting in a sharp increase in core gene families (for example, from 357 at 80% to 1,953 at 0% `length difference cutoff`).

These findings suggest that length differences within gene families significantly influence both the size and structure of the inferred pangenome.

Groups	DNA			AA		
	0% LD	40% LD	80% LD	0% LD	40% LD	80% LD
Core genes	2,626	2,012	432	1,953	1,314	357
Soft core genes	527	1,107	2,121	1,162	1,747	2,139
Shell genes	2,570	2,665	3,185	2,635	2,709	3,220
Cloud genes	14,243	23,320	34,908	22,461	27,941	37,014
Total genes	19,966	29,104	40,646	28,211	33,711	42,730

Table 8: PyamilySeq gene group outputs of different CD-HIT `length difference cutoff's` of 0%, 40% and 80%. Results for both DNA and AA clustering are shown. LD = Length Difference

Although sequence similarity remains central to gene clustering, allowing genes with large length disparities to cluster together can dramatically shift core and accessory gene group composition. This highlights the importance of carefully selecting parameters such as `length difference cutoff` to prevent the misgrouping of phylogenetically or functionally distinct genes. Notably, adjustments to CD-HIT `length difference cutoff` resulted in PyamilySeq drastically increasing its core gene family count to be more inline with what Roary and Panaroo reported. Although these values remain lower than Roary and Panaroo's default estimates, they align with the reported range of the *E. coli* pangenome [30, 31, 32, 33]. However, core and soft core gene counts in past studies have varied widely, possibly due to differences in clustering tools, parameter settings, and core gene definitions. This variability reinforces the need for transparent, flexible, and well-documented clustering methodologies in pangenomic research, ensuring reproducibility and accuracy in gene family classification. To this end, since version 1.5.1, Panaroo now exposes the CD-HIT `length difference cutoff` parameter to address the aforementioned issues of clustering varying-length sequences together. However, this is not the default and requires user understanding and investigation to utilise.

2.7 Reclustering breaks gene clusters

As discussed previously, pangenome inference is only as reliable as the data it uses. The quality and completeness of both genome assemblies and gene annotations directly influence the composition of inferred gene families. In a previous study I showed that even for well-studied bacteria such as *E. coli*, there still exists regions of its genome without annotation [12]. StORF-Reporter was then built to investigate these 'intergenic' regions for the types of genes contemporary annotation tools often miss [18]. Therefore, to asses whether these potentially missing genes sequences are hindering our pangenomc inference, the 74 *E. coli* genomes were re-annotated using StORF-Reporter, supplementing the original Ensembl annotations without modifying them. StORF-Reporter inserts additional CDS predictions (StORFs) into the GFF files, allowing Roary and Panaroo to process them as standard CDS sequences. Given Roary's sensitivity to the `dont split paralogs` option, Table 9 presents the increase in core gene families following StORF-Reporter enhancement.

Gene families	Roary	Roary & StORFs	Roary -s	Roary -s & StORFs
Core genes	1,023	1,202	2,554	2,821
Soft core genes	860	1,238	602	1,076
Shell genes	4,496	9,287	2,430	6,332
Cloud genes	31,890	69,651	16,000	48,021
Total genes	38,469	81,378	21,586	58,250

Table 9: The influence that adding additional StORF sequences has on the output of Roary (processing the 74 *E. coli* dataset). By default Roary is run with 8 CPUs and core alignment turned on. '-s' signifies the use of the `dont split paralogs` option.

Panaroo, in particular, reported a significant increase, with core gene families rising from 2,824 to 3,594. While this might seem like a positive outcome, the results are not entirely straightforward. Ideally, the enhanced dataset should either expand existing clusters or introduce new ones, yet this is not always the case. For example, in the original clustering, group_16792 was classified as a core gene family, but with the additional StORF sequences, it no longer meets the core gene criteria. The Ensembl annotated gene from genome 38344_G02, previously clustered within group_16792, is now isolated as a singleton cluster (group_13871). This shift is unexpected, and in addition, while Roary and PyamilySeq (with 0% `dont split paralogs`) reports the same core gene family, when the additional StORF sequences are introduced, they do not drop this gene in the same way that Panaroo does. This single sequence, ENSB_ru5NaBpXuypHUlL (97 AA), is the only shorter sequence in a family where all other members are 211 AA. Strangely, despite this exclusion, much smaller StORF sequences remain clustered within core families. For instance, in ASM16791v2, the 53 AA StORF sequence ('StORF_62452c9dbe09d477') is grouped with its 211 AA counterpart ('ENSB_12uI2Ow5HXCFra6'). These inconsistencies suggest internal filtering or clustering mechanisms that do not follow an immediately apparent logic, raising questions about how pangenome tools handle alternative annotations and sequence variability.

While the exact mechanisms driving these clustering inconsistencies remain unclear, PyamilySeq provides a way to systematically track the impact of newly annotated sequences without disrupting existing clusters. By holding the original sequences and clusters in place, PyamilySeq allows us to observe where new sequences integrate, offering a controlled approach to evaluating annotation-driven changes. Table 11 summarises

Gene families	Panaroo	Panaroo & StORFs
Core genes	2,824	3,594
Soft core genes	472	956
Shell genes	3,040	6,323
Cloud genes	15,190	38,455
Total genes	21,526	49,328

Table 10: Gene family distribution of the 74 *E. coli* genomes reported by Panaroo with and without the additional StORF sequences.

where the additional StORFs were integrated into existing clusters using PyamilySeq's Partial Mode with the reclustering option. These results demonstrate that while some entirely new core and soft-core gene families emerged from StORF sequences alone, the majority of changes involved existing gene families. Notably, several gene families that were previously classified as non-core transitioned into soft core or core families following the incorporation of StORFs. For a detailed breakdown of each cluster/gene family type, refer to List 1.

2.8 These are not the genes you are looking for

A fundamental assumption in pangenome analysis is that the representative sequences selected for each core gene family accurately reflect the group they represent. Roary, Panaroo, and PyamilySeq all report such representative sequences, intended to serve as a centroid, the most typical or central sequence within a gene family. However, as we might now expect, this does not always hold true. Examining the gene_presence_absence files reveals clear instances where representative sequences are misleading. In the Panaroo (refound-off - to ensure we are not getting sequences not in the provided annotations) analysis of the 74 *E. coli* genomes, core gene family group_15420 consists of 74 sequences, 73 of which are 564 nt / 188 AA in length. Yet, its assigned representative sequence is 1,494 nt / 498 AA, nearly three times longer. This arises because Roary and Panaroo select the longest sequence by default, rather than the most representative. The 73 shorter sequences align perfectly (100% pident) with a known membrane integrity-associated transporter subunit PqiC (WP_000759128.1, 187 AA). However, the single longer sequence aligns instead with a paraquat-inducible protein B (STK76389.1, 497 AA), a different gene with a distinct function. This means that for group_15420, the assigned representative sequence does not match the actual gene or function of the majority of its members. This is not just a minor inconsistency, it fundamentally undermines downstream functional analysis. If a researcher used these sequences for comparative genomics, pathway reconstruction, or functional annotation, they would be working with an incorrect sequence and thus outcome.

Gene Groups	**80% / [0%]**
First_core_99	357 / [**1953**]
First_core_95	2139 / [**1162**]
First_core_15	3219 / [**2633**]
First_core_0	37014 / [**22463**]
extended_core_99	0 / [**28**]
extended_core_95	17 / [**31**]
extended_core_15	95 / [**217**]
extended_core_0	264 / [**401**]
combined_core_99	0 / [**3**]
combined_core_95	0 / [**0**]
combined_core_15	8 / [**7**]
combined_core_0	4 / [**2**]
Second_core_99	0 / [**0**]
Second_core_95	1 / [**3**]
Second_core_15	41 / [**76**]
Second_core_0	356 / [**1006**]
only_Second_core_99	136 / [**149**]
only_Second_core_95	494 / [**523**]
only_Second_core_15	4030 / [**3948**]
only_Second_core_0	38443 / [**36133**]
# FCs (Incl. Singletons)	42730 / [**28211**]
# SCs (Incl. Singletons)	85824 / [**68954**]
# FCs (w/ Add' Second Seqs, No New Genomes)	17 / [**404**]

Table 11: Detailed summary of gene family/group classifications reported by reclustering StORF sequences with PyamilySeq under two parameter sets: 80% and 0% `length difference cutoff` (values in square brackets). Each row represents a category of gene families based on their occurrence and relationship across the original (First) and newly added (Second) sequences. The final three rows summarise total gene family counts: # FCs (Incl. Singletons): Total number of First gene families including singletons. # SCs (Incl. Singletons): Total number of Second gene families including singletons. # FCs (w/ Add'l Second Seqs, No New Genomes): Number of First_core families that were expanded by additional Second sequences, but did not include any newly added genomes.

Once again, this highlights the hidden pitfalls of blindly using complex tools, where small, often overlooked algorithmic choices introduce systematic errors that can make results unusable for biological interpretation. These are not the genes you are looking for.

3 Discussion and Conclusion

The study of pangenomics plays a critical role in understanding microbial evolution, functional diversity, and the distribution of genes across a taxonomic range, yet the accuracy of its insights is fundamentally dependent on the methodologies used. This study highlights multiple sources of variation and bias in gene clustering and pangenome inference, demonstrating how parameter selection and seemingly arbitrary algorithmic decisions can significantly reshape the reported composition of core and accessory gene families. For example, most sequence alignment or clustering tools, including BLAST (BLAST-N for DNA, BLAST-P for AA) and CD-HIT (CD-HIT-EST for DNA clustering), operate under the assumption that codon translation follows the universal codon table 11, applying it uniformly across genomes. However, this overlooks non-standard codons and alternative genetic codes, which, though rare, are present in certain taxa (e.g., *Mycoplasma*). Panaroo partially addresses this by clustering 'concomitant genes at the DNA level', yet its overall approach still relies heavily on homologous signal in AA sequences. To systematically investigate potential biases, PyamilySeq was used to infer gene families from both DNA and AA sequence clustering, employing CD-HIT and CD-HIT-EST accordingly. While pangenomic tools such as Roary and Panaroo have been instrumental in advancing the field, they are not immune to these methodological artifacts, and their influence on downstream analyses is often underestimated.

Another key issue uncovered in this study is the assumption of determinism in bioinformatic tools. Many researchers operate under the assumption that running the same tool with broadly the same parameters will yield consistent results. However, this study demonstrates that factors as seemingly trivial as CPU and memory allocation, floating point parameter precision, and the length difference cutoff in CD-HIT can all lead to drastically different gene family assignments. Furthermore, the lack of transparency in sequence clustering and pangenome tools has led to unexpected issues. As we discovered, requesting an alignment of core gene sequences can inexplicably alter gene family composition. More concerningly, representative sequences for core gene families are potentially selected based on length rather than biological relevance, leading to incorrect functional conclusions. Such inconsistencies raise questions about the reproducibility and reliability of pangenomic studies, particularly when their conclusions influence fields such as antimicrobial resistance surveillance and pathogen detection.

Responsibility for the issues raised in this study is shared widely across the bioinformatics community. Although concerns over software quality in bioinformatics have been recognised for some time [34], the development and maintenance of computational tools are often underfunded and largely driven by small teams or even solo developers, who lack the resources to support their tools long term. Compounding this, there is widespread misunderstanding and misuse of tools within the community. Software designed for a narrow purpose is frequently repurposed for broader applications, sometimes becoming overextended into multi-functional "Swiss army knives", far beyond the original intent of their developers. Furthermore, papers describing novel tools are static snapshots, and documentation is often limited or quickly outdated. Many developers are domain scientists rather than trained software engineers and rely heavily on their user communities to report bugs or inconsistencies. Yet, with a user base that spans a broad range of expertise, it can be overwhelming, even for experienced users, to identify, understand, and communicate these problems effectively. The result is a complex and decentralised ecosystem where small errors can propagate widely, and the line between biological insight and computational artefact becomes increasingly blurred.

Additionally, while the biological basis of gene clustering relies on sequence similarity, this study reveals that high similarity alone does not always justify grouping genes together. Gene birth [35], neofunctionalisation [36], and pseudogenisation [37] introduce complexity that is difficult, if not impossible, to account for in current clustering approaches. As demonstrated, sequences of vastly different lengths can be grouped within the same family, while near-identical sequences may be excluded due to arbitrary rules. Panaroo's 'refinding' system, which attempts to recover missing genes using BLAST, represents an important step toward mitigating these issues. However, the extent to which this system influences final gene family assignments is not always transparent, making it difficult for researchers to assess its impact on their results.

To address these ongoing challenges, PyamilySeq was developed as both a diagnostic tool to investigate biases in gene clustering and a prac-

tical framework for refining pangenome inference. By allowing researchers to systematically explore the impact of clustering parameters and grouping thresholds, PyamilySeq provides an unparalleled level of transparency compared to existing tools. Unlike rigid, almost black-box approaches, it enables iterative reclustering, allowing newly annotated sequences to be incorporated without disrupting existing clusters. Furthermore, the PyamilySeq toolkit is presented with several additional tools that can facilitate the exploration of previous clustering or pangenome results to help researchers understand any potential problems or oddities that may exist.

Despite these advantages, PyamilySeq is not intended to replace contemporary techniques but to complement them by exposing their consequences and guiding the development of more transparent methodologies. It underscores the need for greater user awareness of the impact of methodological choices, reinforcing the importance of parameter sensitivity analysis in gene clustering and pangenomics. Future efforts should focus on integrating biologically informed clustering strategies that account for evolutionary processes beyond sequence similarity alone. By critically evaluating the tools that define pangenomes, we can move toward more accurate, reproducible, and biologically meaningful genomic insights.

4 Methods

4.1 Data selection and use

4.1.1 10 *Escherichia coli* genome dataset

The first dataset used in this study was a set of 10 *E. coli* genomes, which varied widely in genome fragmentation (1 to 412 contigs) and in the number of coding genes (4,540 to 5,565). This collection of genomes was selected and downloaded from release 59 of Ensembl Bacteria [38]. This 'diversity' of assembly quality and gene number allowed for an assessment of PyamilySeq's performance and utility on a small dataset with varying levels of genome completeness and annotation quality (see Supplementary Table 1 for more details).

For sequence clustering, CD-HIT was used with its default parameters of a sequence identity threshold of **0.90** and a length difference threshold of **0.80**. Analysis was performed on both the DNA and AA sequences and is reported separately. In parallel, pangenome analysis was performed on the same dataset using both Roary and Panaroo. Please see Subsection 4.2 for information regarding runtime parameters.

4.1.2 74 *Escherichia coli* genome dataset

To demonstrate a more representative use-case for a pangenome study, a larger collection of *E. coli* genomes was downloaded from Ensembl Bacteria (release 59). In an attempt to select for 'high-quality' genomes, those with more than five fragments (excluding plasmids) were filtered out. This filtering step aimed to only include those genomes with a higher completeness and reduce the number of potential assembly errors.

Furthermore, to verify taxonomic assignments, knowing that incorrect assignments can lead to underestimating a pangenomic core, the genomes were analysed using GTDB-TK [39], which reclassified three of the genomes initially labelled as *Escherichia coli* as belonging to the genus *Citrobacter*. Following this refinement, a final dataset of 74 genomes was collected and processed with default parameters, with further details available in Supplementary Table 2.

4.2 Parameter selection

Unless where otherwise specified, all tools in this study were used with the follow parameters.

Roary v3.13.0 parameters: All parameters were set at defaults apart from: CPU allocation at 8 **(-p 8)** and create a multifasta alignment with MAFFT **(-e -n)**.

Panaroo v1.5.1 parameters: All parameters were set at defaults apart from: CPU allocation at 8 **(-t 8)**, stringency mode of moderate **(–clean-mode moderate)** and create a core gene alignment **(-a core)**.

PyamilySeq v1.x.x parameters: All parameters were set at defaults apart from: CPU allocation at 8 **(-T 8)**, Memory allocation of 4,000MBs **(-M 4000)**, a CD-HIT identity threshold of 90% **(-c 0.90)**, a length difference cutoff of 0.80 **(0.80)** and create a core gene alignment **(-a core)**. Additionally, PyamilySeq has been designed to expand all single decimal place parameters to two decimal places (0.0 to 0.00).

4.3 The PyamilySeq toolkit

PyamilySeq is a modular framework developed to both generate pangenomes and systematically investigate the sequence clustering and inference steps that underpin them. It includes a suite of tools that replicate and dissect key stages in typical pangenome workflows, such as gene clustering, core/accessory gene classification, and alignment. This enables users to examine how parameter choices and algorithmic behaviour influence outcomes. By providing transparency at each stage, PyamilySeq allows researchers to better understand the assumptions, limitations, and variability inherent in widely used pangenomic tools.

PyamilySeq offers a high level of customisation and user control, distinguishing it from many existing tools that limit users to predefined clustering methods and parameters. Users can perform initial clustering externally, with a method of choice, including but not limited to: CD-HIT, BLAST, DIAMOND [40], or MMseqs2 [41], and import the clustering output into PyamilySeq for pangenomic analysis. PyamilySeq does this by supporting the .clstr format of CD-HIT and any tab-separated file (formatted as *Node1* \t *Node2*), which represents the relationships between gene sequences or clusters. This flexibility, supported by PyamilySeq's ability to handle both DNA and amino acid (AA) sequences and non-coding genes, allows researchers to tailor their analysis to various genomic contexts. The tool also provides customisable definitions for core, soft-core, and accessory gene groups, allowing users to align these classifications with their dataset's biological distribution.

PyamilySeq provides various outputs including; a *summary_statistics.txt* file that follows a format similar to those reported by Roary and Panaroo - with additional modifications reflecting the extended functionality of PyamilySeq (see List 1), a presence-absence file matching the format used by Roary for compatibility with existing downstream analysis tools, individual and combined FASTA files containing the gene sequences of each identified gene family and a core-gene alignment using MAFFT [19]. By offering this level of customisation, PyamilySeq ensures that users have near-full control over the clustering process, allowing for more meaningful and accurate results tailored to the needs of the study

The PyamilySeq toolkit consists of several tools that can be used independently or together, depending on the input data and the type of analysis undertaken. The following sections describe the main tools along with the data formats they take or create, all of which are available both on GitHub (`https://github.com/NickJD/PyamilySeq`) and as part of the unified PyamilySeq PyPi package (`https://pypi.org/project/PyamilySeq/`).

PyamilySeq operates in two primary modes: Full Mode and Partial Mode - see Figure 9.

```
usage: PyamilySeq [-h] {Full,Partial} ...

PyamilySeq v1.1.2: A tool for gene clustering and analysis.

positional arguments:
  {Full,Partial}  Choose a mode: 'Full' or 'Partial'.
    Full          Full mode: PyamilySeq to cluster with CD-HIT and process
                  output.
    Partial       Partial mode: PyamilySeq to process pre-clustered data.

options:
  -h, --help      show this help message and exit
```

Figure 9: PyamilySeq menu system showing the two main options of Full and Partial modes.

4.3.1 PyamilySeq: Full Mode

Full Mode in PyamilySeq provides a complete gene clustering and pangenome analysis pipeline, automating the processing of annotated genomes to generate gene families and pangenome statistics. This mode is designed for users starting with annotated GFF files and produces a gene presence-absence matrix by default. The workflow broadly consists of the following steps:

1. **Input:** PyamilySeq reads genome data from a directory, accepting either combined GFF+FASTA files or separate GFF and FASTA inputs. Users can specify which genomic features to extract (e.g., CDS, rRNA, pseudogenes), ensuring compatibility with various annotation formats.

2. **Clustering:** Genes are clustered into families using CD-HIT, with user-definable thresholds for sequence identity and length difference. Clustering can be performed in either nucleotide or amino acid space, depending on the user's preference.

3. **Pangenome Inference:** Gene families are classified as core, soft-core, or accessory genes based on customisable presence thresholds across genomes.

4. **Gene Presence-Absence Matrix:** PyamilySeq generates a matrix showing the distribution of gene families across genomes, formatted for downstream visualisation and comparative analyses.

5. **Optional Core Genome Alignment:** Representative core genes can be aligned using MAFFT, enabling concatenated sequence generation for phylogenetic or comparative studies.

Full Mode is ideal for researchers seeking an end-to-end solution for pangenome analysis, providing transparent gene clustering, classification, and phylogenetic insights directly from annotated genome data.

4.3.2 PyamilySeq: Partial Mode

Partial Mode provides greater flexibility by allowing users to input pre-clustered gene sets instead of running CD-HIT clustering within PyamilySeq. This mode is designed for cases where external tools, custom clustering methods, or specific parameters have already been applied, and users want PyamilySeq to perform the final stages of pangenome analysis. The workflow consists of:

1. **Input:** Users provide pre-clustered gene sets in either CD-HIT format (.clstr file) or as an edge-list file (`Node1 \t Node2`), ensuring compatibility with alternative clustering methods such as MMseqs2, DIAMOND, or BLAST.

2. **Pangenome Inference:** PyamilySeq processes the pre-clustered data to classify genes into core, soft-core, and accessory categories, applying the same customisable presence thresholds as in Full Mode.

3. **Gene Presence-Absence Matrix:** A presence-absence matrix is generated from the provided clusters, formatted for downstream visualisation and comparative analysis.

4. **Optional Core Genome Alignment:** If selected, PyamilySeq aligns representative core genes using MAFFT, following the same approach as Full Mode.

Partial Mode is ideal for researchers who have already performed clustering using external tools and want PyamilySeq to handle pangenome inference, matrix generation, and optional alignment. Additionally, Seq-Combiner can be used to prepare a directory of GFF/FASTA files with the appropriate sequence IDs (see Supplementary Subsection 'Input handling with Seq-Combiner' for details).

4.3.3 PyamilySeq: Reclustering

To support ongoing genome annotation research, PyamilySeq includes a **reclustering** feature that allows newly annotated gene sequences to be integrated into existing clusters. This is especially useful when additional sequences become available after initial clustering, enabling **continuous refinement** of gene families. PyamilySeq reassesses the position of these new sequences in the context of the original clustering, determining whether they fit within established clusters or if they form new ones. This approach not only improves the accuracy of gene family, but also reveals previously undetected genetic elements that may have been overlooked in the initial analysis.

Reclustering is available in Partial Mode, allowing users to **update and refine pangenome inferences without reprocessing entire datasets**. In Partial Mode, PyamilySeq can recluster additional sequences within an already clustered dataset, making it especially useful for incorporating newly annotated gene types (e.g., pseudogenes, alternative ORFs) and evaluating their impact on gene grouping.

A key example of this approach can be seen in the StORF-Reporter study [18], which demonstrated how the integration of StORFs (Stop-Open Reading Frames), often longer than standard gene annotations, had a significant impact on gene clustering. Due to their extended lengths, StORFs had the potential to either **merge or fragment** existing gene groups. In that study, accurately reporting the true distribution of these newly annotated sequences required a **reclustering approach**, a feature absent in other tools. Figures 10 and 11 illustrate how the addition of longer sequences can dramatically shift clustering results.

By providing a way to incorporate novel gene predictions and alternative gene types, such as pseudogenes and StORFs, into established pangenome frameworks, PyamilySeq's reclustering function offers a powerful tool for researchers. It allows for an iterative refinement of gene classifications, revealing how new sequences influence the structure of core and accessory gene families, ultimately leading to a more biologically meaningful representation of microbial genomes.

5 Software versions used

```
PyamilySeq: v1.1.2
CD-HIT: v4.8.1
```

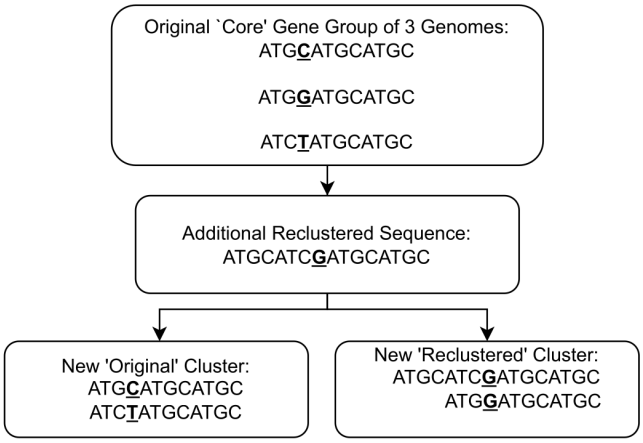

Figure 10: Reclustering additional sequences can fragment original clusters, decreasing the number of core/soft-core gene groups.

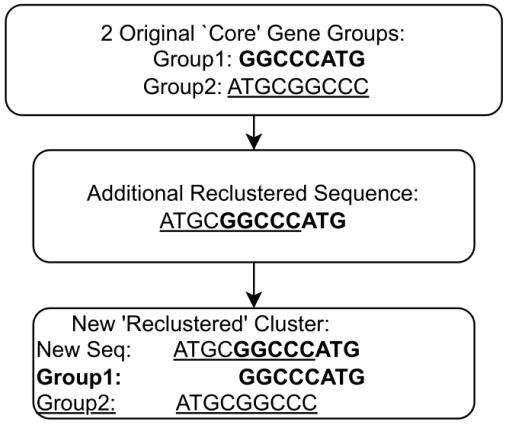

Figure 11: Reclustering additional sequences can merge existing clusters, changing the overall number of core/soft-core gene groups.

```
Panaroo: v1.5.1
Roary: v3.13.0
MAFFT: v7.490
Fasttree: v2.1.11
ETE 3: v3.1.3
clustalo: v1.2.4
mview: v1.67
```

6 Competing interests

No competing interest are declared.

7 Author contributions statement

N.J.D undertook all the development and testing of PyamilySeq, investigated the clustering and pangenome tools studied in this work and wrote the manuscript.

8 Acknowledgments

N.J.D. would like to thank Prof. Michael Surette and Dr. Katie Lawther for their significant advice and guidance, which has greatly improved the manuscripts and PyamilySeqs' applicability to a number of biological questions. Additionally, N.J.D would like to acknowledge Dr. Amanda Clare and Dr. Fiona Whelan for their comments and support of the project.

References

[1] Jiannan Chao, Furong Tang, and Lei Xu. Developments in algorithms for sequence alignment: A review. *Biomolecules*, 12(4):546, 2022.

[2] Hervé Tettelin, Vega Masignani, Michael J Cieslewicz, Claudio Donati, Duccio Medini, Naomi L Ward, Samuel V Angiuoli, Jonathan Crabtree, Amanda L Jones, A Scott Durkin, et al. Genome analysis of multiple pathogenic isolates of streptococcus agalactiae: implications for the microbial "pan-genome". *Proceedings of the National Academy of Sciences*, 102(39):13950–13955, 2005.

[3] Ming-Ren Yang, Shun-Feng Su, and Yu-Wei Wu. Using bacterial pan-genome-based feature selection approach to improve the prediction of minimum inhibitory concentration (mic). *Frontiers in Genetics*, 14:1054032, 2023.

[4] Jason W Shapiro and Catherine Putonti. Rephine. r: a pipeline for correcting gene calls and clusters to improve phage pangenomes and phylogenies. *PeerJ*, 9:e11950, 2021.

[5] Saioa Manzano-Morales, Yang Liu, Sara González-Bodí, Jaime Huerta-Cepas, and Jaime Iranzo. Comparison of gene clustering criteria reveals intrinsic uncertainty in pangenome analyses. *Genome Biology*, 24(1):250, 2023.

[6] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[7] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[8] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):2542, 2018.

[9] GS Vernikos. A review of pangenome tools and recent studies. *The pangenome: diversity, dynamics and evolution of genomes*, pages 89–112, 2020.

[10] Aysun Urhan and Thomas Abeel. A comparative study of pan-genome methods for microbial organisms: Acinetobacter baumannii pan-genome reveals structural variation in antimicrobial resistance-carrying plasmids. *Microbial Genomics*, 7(11):000690, 2021.

[11] Steven L Salzberg. Next-generation genome annotation: we still struggle to get it right. *Genome biology*, 20(1):92, 2019.

[12] Nicholas J Dimonaco, Wayne Aubrey, Kim Kenobi, Amanda Clare, and Christopher J Creevey. No one tool to rule them all: prokaryotic gene prediction tool annotations are highly dependent on the organism of study. *Bioinformatics*, 38(5):1198–1207, 2022.

[13] Lisa R Moore, Ron Caspi, Dana Boyd, Mehmet Berkmen, Amanda Mackie, Suzanne Paley, and Peter D Karp. Revisiting the yome of escherichia coli. *Nucleic Acids Research*, 52(20):12201–12207, 2024.

[14] Sung-Huan Yu, Jörg Vogel, and Konrad U Förstner. Annogesic: a swiss army knife for the rna-seq based annotation of bacterial/archaeal genomes. *Gigascience*, 7(9):giy096, 2018.

[15] Gerry Tonkin-Hill, Neil MacAlasdair, Christopher Ruis, Aaron Weimann, Gal Horesh, John A Lees, Rebecca A Gladstone, Stephanie Lo, Christopher Beaudoin, R Andres Floto, et al. Producing polished prokaryotic pangenomes with the panaroo pipeline. *Genome biology*, 21:1–21, 2020.

[16] Andrew J Page, Carla A Cummins, Martin Hunt, Vanessa K Wong, Sandra Reuter, Matthew TG Holden, Maria Fookes, Daniel Falush, Jacqueline A Keane, and Julian Parkhill. Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics*, 31(22):3691–3693, 2015.

[17] Kevin Lamkiewicz, Lisa-Marie Barf, Konrad Sachse, and Martin Hölzer. Ribap: a comprehensive bacterial core genome annotation pipeline for pangenome calculation beyond the species level. *Genome Biology*, 25(1):170, 2024.

[18] Nicholas J Dimonaco, Amanda Clare, Kim Kenobi, Wayne Aubrey, and Christopher J Creevey. StORF-Reporter: finding genes between genes. *Nucleic Acids Research*, 51(21):11504–11517, 2023.

[19] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, 30(14):3059–3066, 2002.

[20] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. FastTree 2–approximately maximum-likelihood trees for large alignments. *PloS one*, 5(3):e9490, 2010.

[21] David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1-2):131–147, 1981.

[22] Jaime Huerta-Cepas, François Serra, and Peer Bork. Ete 3: reconstruction, analysis, and visualization of phylogenomic data. *Molecular biology and evolution*, 33(6):1635–1638, 2016.

[23] Carlos P Cantalapiedra, Ana Hernández-Plaza, Ivica Letunic, Peer Bork, and Jaime Huerta-Cepas. eggnog-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. *Molecular biology and evolution*, 38(12):5825–5829, 2021.

[24] Ari Löytynoja. Phylogeny-aware alignment with prank. In *Multiple sequence alignment methods*, pages 155–170. Springer, 2013.

[25] Nick Dimonaco. *ORFs, StORFs and Pseudogenes: Uncovering Novel Genomic Knowledge in Prokaryotic and Viral Genomes*. PhD thesis, Aberystwyth University, 2022.

[26] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.

[27] Fábio Madeira, Nandana Madhusoodanan, Joonheung Lee, Alberto Eusebi, Ania Niewielska, Adrian RN Tivey, Rodrigo Lopez, and Sarah Butcher. The embl-ebi job dispatcher sequence analysis tools framework in 2024. *Nucleic acids research*, 52(W1):W521–W525, 2024.

[28] Ariel Hecht, Jeff Glasgow, Paul R Jaschke, Lukmaan A Bawazer, Matthew S Munson, Jennifer R Cochran, Drew Endy, and Marc Salit. Measurements of translation initiation from all 64 codons in e. coli. *Nucleic acids research*, 45(7):3615–3626, 2017.

[29] José Luis López, Mauricio Javier Lozano, María Laura Fabre, and Antonio Lagares. Codon usage optimization in the prokaryotic tree of life: how synonymous codons are differentially selected in sequence domains with different expression levels and degrees of conservation. *MBio*, 11(4):10–1128, 2020.

[30] David A Rasko, MJ Rosovitz, Garry SA Myers, Emmanuel F Mongodin, W Florian Fricke, Pawel Gajer, Jonathan Crabtree, Mohammed Sebaihia, Nicholas R Thomson, Roy Chaudhuri, et al. The pangenome structure of escherichia coli: comparative genomic analysis of e. coli commensal and pathogenic isolates. *Journal of bacteriology*, 190(20):6881–6893, 2008.

[31] Miriam Land, Loren Hauser, Se-Ran Jun, Intawat Nookaew, Michael R Leuze, Tae-Hyuk Ahn, Tatiana Karpinets, Ole Lund, Guruprased Kora, Trudy Wassenaar, et al. Insights from 20 years of bacterial genome sequencing. *Functional & integrative genomics*, 15:141–161, 2015.

[32] Arun Gonzales Decano and Tim Downing. An escherichia coli st131 pangenome atlas reveals population structure and evolution across 4,071 isolates. *Scientific reports*, 9(1):17394, 2019.

[33] Zhi-Kai Yang, Hao Luo, Yanming Zhang, Baijing Wang, and Feng Gao. Pan-genomic analysis provides novel insights into the association of e. coli with human host and its minimal genome. *Bioinformatics*, 35(12):1987–1991, 2019.

[34] Brendan Lawlor and Paul Walsh. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. *Bioengineered*, 6(4):193–203, 2015.

[35] Stephen Branden Van Oss and Anne-Ruxandra Carvunis. De novo gene birth. *PLoS genetics*, 15(5):e1008160, 2019.

[36] Kosuke M Teshima and Hideki Innan. Neofunctionalization of duplicated genes under the pressure of gene conversion. *Genetics*, 178(3):1385–1398, 2008.

[37] Emmanuelle Lerat and Howard Ochman. Recognizing the pseudogenes in bacterial genomes. *Nucleic acids research*, 33(10):3125–3132, 2005.

[38] Andrew D Yates, James Allen, Ridwan M Amode, Andrey G Azov, Matthieu Barba, Andrés Becerra, Jyothish Bhai, Lahcen I Campbell, Manuel Carbajo Martinez, Marc Chakiachvili, et al. Ensembl genomes 2022: an expanding genome resource for non-vertebrates. *Nucleic acids research*, 50(D1):D996–D1003, 2022.

[39] Pierre-Alain Chaumeil, Aaron J Mussig, Philip Hugenholtz, and Donovan H Parks. Gtdb-tk: a toolkit to classify genomes with the genome taxonomy database, 2020.

[40] Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nature methods*, 18(4):366–368, 2021.

[41] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.