Flawed machine-learning confounds coding sequence annotation

DJ Champion[1], Ting-Hsuan Chen[2], Susan Thomson[2], Michael A. Black[1], Paul P. Gardner[1]*

Abstract

Detecting protein coding genes in genomic sequences is a significant challenge for understanding genome functionality, yet the reliability of bioinformatic tools for this task remains largely unverified. This is despite some of these tools having been available for several decades, and being widely used for genome and transcriptome annotation.

We perform an assessment of nucleotide sequence and alignment-based *de novo* protein-coding detection tools. The controls we use exclude any previous training dataset and include coding exons as a positive set and length-matched intergenic and shuffled sequences as negative sets.

Our work demonstrates that several widely used tools are neither accurate nor computationally efficient for the protein-coding sequence detection problem. In fact, just three of nine tools significantly outperformed a naive scoring scheme. Furthermore, we note a high discrepancy between self-reported accuracies and the accuracy achieved in our study. Our results show that the extra dimension from conserved and variable nucleotides in alignments have a significant advantage over single sequence approaches.

These results highlight significant limitations in existing protein-coding annotation tools that are widely used for lncRNA annotation. This shows a need for more robust and efficient approaches to training and assessing the performance of tools for identifying protein-coding sequences. Our study paves the way for future advancements in comparative genomic approaches and we hope will popularise more robust approaches to genome and transcriptome annotation.

[1] *Department of Biochemistry, University of Otago, Dunedin, New Zealand.*

[2] *The New Zealand Institute for Plant and Food Research Limited, Lincoln, New Zealand.*

***Corresponding author**: paul.gardner@otago.ac.nz

Introduction

Annotating protein-coding regions within genomes and transcripts remains a crucial bioinformatic challenge [1, 2, 3], particularly in the context of differentiating between protein-coding and non-coding transcripts for analyzing nucleotide sequences [4, 5]. However, assessing how well preferred tools handle realistic datasets, which include truncations, sequencing, and processing errors, remains unclear [3].

In model species like *Homo sapiens*, *Mus musculus*, *Drosophila melanogaster*, and *Saccharomyces cerevisiae*, rigorous manual biocuration has significantly refined genome annotations [6, 7, 8]. For example, *H. sapiens* genome annotations have evolved from an initial estimate of 30,000 protein-coding genes to about 20,000 through decades of meticulous review, which highlights the importance of cautious interpretation of predicted annotations [9]. These annotations have been enhanced by experimental methods that detect active gene expression in specific tissues and conditions, though such methods are limited by experimental parameters and can include noisy non-functional transcription and translation signals [9, 3, 10, 11, 12].

As genome sequencing becomes more accessible, manual annotation, once considered the gold standard, has given way to automated techniques for annotating both genomes and transcriptomes [13, 3, 14, 15]. Despite extensive literature on genome annotation [16, 5, 17, 18, 2], there are few systematic evaluations of tools that differentiate coding from non-coding sequences [1, 13, 17, 19, 20]. Therefore, we think it is timely for further evaluations of genome annotation tools.

The field of genome annotation could greatly benefit from unbiased benchmarking protocols similar to those used in protein structure prediction, such as those in the Critical Assessment of Protein Structure Prediction (CASP) initiative [21]. CASP has driven advancements by promoting rigorous data collection and innovation, leading to high-accuracy tools like AlphaFold2 and RosettaFold [22, 23]. Adopting a similar framework could significantly enhance the accuracy and reliability of genome annotation tools, improving our understanding of genome function.

Software benchmarks are subject to many caveats that are the result of inevitable limitations. In spite of these caveats benchmarks serve a valuable purpose in providing an assessment of how tools perform on given datasets at a particular point in time. They can also identify performance limitations and directions where further improvements may be made. In brief, we: 1. use default parameters in each case in order to avoid inflating individual tool performance [24], 2. we may apply some tools in a manner in which they were not designed (e.g. a transcriptome tool can be applied to genomic sequences), and 3. some of the control data may be mislabelled, leading to conflicted outcomes. These caveats have been documented in detail elsewhere [25].

In this study, we evaluate the effectiveness of *de novo* coding annotation tools for eukaryotic nucleotide sequences that do not need specific training. These tools differentiate between nucleotide sequences or alignments that are either constrained by genetic codes or are non-coding. Given the distinct statistical signatures that coding sequences typically exhibit compared to non-coding ones, this task should be feasible using bioinformatic methods. The findings of this research have significant implications for the long non-coding RNA (lncRNA) research community, which heavily relies on these tools to identify non-coding transcripts [26].

Methods

Our methodology is divided into these key sections: (1) Dataset preparation, where we describe the acquisition and preparation of coding and non-coding sequences used as controls; (2) Performance measures, where we explain the methods used to analyze the results and verify the significance of our findings. (3) Tool inclusion, detailing the criteria for selecting the annotation tools evaluated in this study; and (4) Benchmarking Strategy, outlining the protocols for tool performance assessment, including accuracy, efficiency, and computational demand; These ensure a comprehensive evaluation of each tools' capabilities in distinguishing between coding from non-coding genomic sequences.

Data selection: positive & negative control sequences

To improve the assessment of software predictions for protein-coding nucleotide sequences and mitigate biases, we avoid commonly used reference genomes such as *Homo sapiens* and *Mus musculus*. We selected representative species from three eukaryotic clades (mammals, plants, and fungi) that have not previously been used in software training. These are *Felis catus* (house cat), *Cucumis melo* (melon), and *Aspergillus puulaauensis*. Further details are available in Supplementary Tables S2 and S3 and Figure 1.

Up to ten further genomes that span a range of phylogenetic distances are selected for each group, these are used to construct multiple sequence alignments for the comparative tools (Table 1). Each selected genome assembly has a high BUSCO completeness score [27, 28] (Supplementary Table S3), and have numbers of annotated coding genes that is consistent with related genomes.

The positive control coding sequences are derived from GenBank annotations. We collapsed overlapping annotations and randomly sampling 1,200 coding exons from each reference genome. Further realism is introduced into the test dataset by including 75 nucleotides of intron or UTR from both sides of each exons (refer to Figure 1A). Negative control intergenic

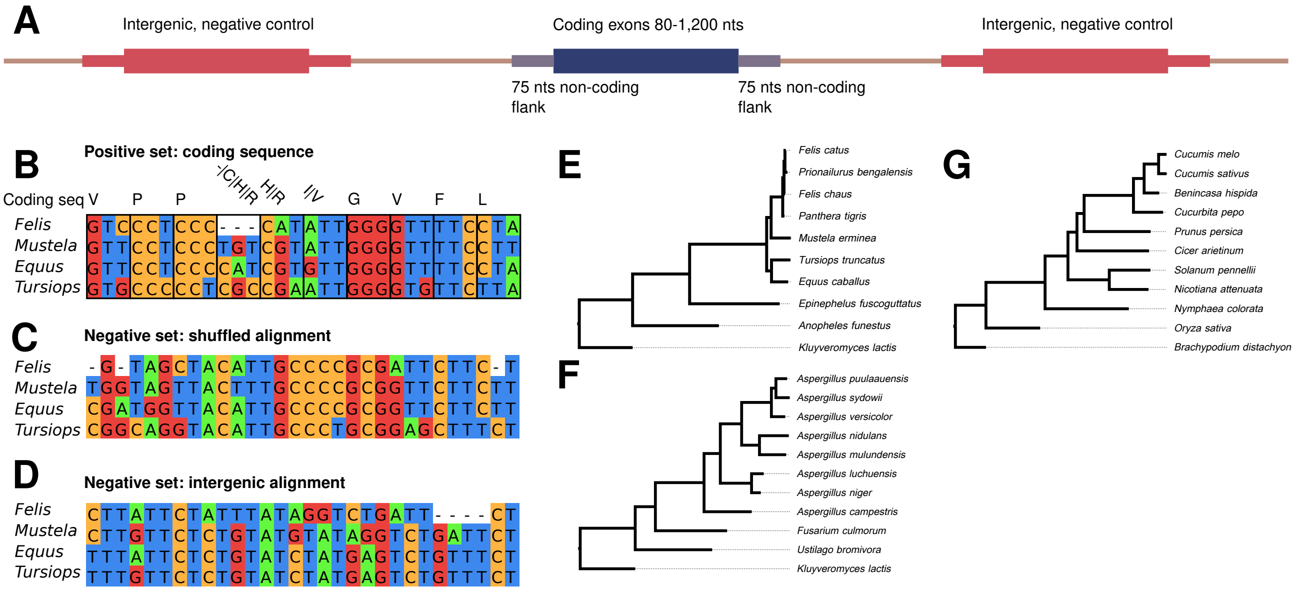

A
Intergenic, negative control
Coding exons 80-1,200 nts
Intergenic, negative control
75 nts non-coding flank
75 nts non-coding flank
B
Positive set: coding sequence
Coding seq V P P -/C/H/R H/R I/V G V F L
Felis GTCCCTCCC---CATATTGGGGTTTTCCTA
Mustela GTTCCTCCCTGTCGTATTGGGGTTTTCCTT
Equus GTTCCTCCCCATCGTGTTGGGGTTTTCCTA
Tursiops GTGCCCCCTCGCCGAATTGGGGTGTTCTTA
C
Negative set: shuffled alignment
Felis -G-TAGCTACATTGCCCCGCGATTCTTC-T
Mustela TGGTAGTTACTTTGCCCCGCGGTTCTTCTT
Equus CGATGGTTACATTGCCCCGCGGTTCTTCTT
Tursiops CGGCAGGTACATTGCCCTGCGGAGCTTTCT
D
Negative set: intergenic alignment
Felis CTTATTCTATTTATAGGTCTGATT----CT
Mustela CTTGTTCTCTGTATGTATAGGTCTGATTCT
Equus TTTATTCTCTGTATCTATGAGTCTGTTTCT
Tursiops TTTGTTCTCTGTATCTATGAGTCTGTTTCT
E
Felis catus
Prionailurus bengalensis
Felis chaus
Panthera tigris
Mustela erminea
Tursiops truncatus
Equus caballus
Epinephelus fuscoguttatus
Anopheles funestus
Kluyveromyces lactis
F
Aspergillus puulaauensis
Aspergillus sydowii
Aspergillus versicolor
Aspergillus nidulans
Aspergillus mulundensis
Aspergillus luchuensis
Aspergillus niger
Aspergillus campestris
Fusarium culmorum
Ustilago bromivora
Kluyveromyces lactis
G
Cucumis melo
Cucumis sativus
Benincasa hispida
Cucurbita pepo
Prunus persica
Cicer arietinum
Solanum pennellii
Nicotiana attenuata
Nymphaea colorata
Oryza sativa
Brachypodium distachyon

Figure 1. Controls for benchmarking protein coding detection tools. A: Annotated exons of 80 to 1,200 nucleotides (nts) are selected from reference genomes, flanking 75 nts are also included with these positive control sequences. These are length-matched with genomically nearby intergenic sequences which serve as one source of negative controls. Shuffled coding sequences are used as a further source of negative control. **B-D:** Exemplar multiple sequence alignments, these include a partial exon (**B**), with corresponding negative controls from a shuffled alignment (**C**) and a neighbouring intergenic region (**D**). For alignment **B** the corresponding amino-acid(s) of each codon is shown along the top line. **E-G:** Phylogenetic trees for genomic reference and selected related genomes for producing alignments, these are the (**E**) animalia (reference: *Felis catus*), (**F**) fungi (reference: *Aspergillus puulaauensis*), and (**G**) plants (reference: *Cucumis melo*).

sequences of the same lengths are selected from adjacent upstream and downstream intergenic regions that exceed one kilobase (kb) in length (for summary statistics see Supplementary Table S3). The intergenic sequences are compared with protein in the UniProt database and possible coding sequences (e-value $< 10^{-10}$, bitscore > 31) were removed using translated searches with MMseqs2 (v15.6f452) [29].

Alignments for comparative tools were made for intergenic and coding sequences (Figure 1). Reference sequences are queried with MMseqs2, and top-scoring matches for each genome selected for multiple sequence alignment with Clustal Omega (1.2.4) [30]. A further negative control set is produced by shuffling coding alignments with `esl-shuffle` (Easel 0.48 (Nov 2020)).

Each of the selected prediction tools were run on both the coding and non-coding sequences or alignments. Subsequently, the optimal coding potential scores for each input for each tool were used as a basis for computing performance statistics, as defined in the below 'Performance metrics' section. For the tools that only scan the forward strand, the reverse complement of each input sequence was also screened. In instances where multiple scores are returned for a sequence (in any of the six-frames), the best score is selected (highest or lowest, depending on which is a better indicator for a coding sequence).

Performance metrics and output management

Sequences are ranked by prediction score, and a sliding threshold is used. The coding sequences that score at or above a threshold are labelled “true positives” (TP), negative control sequences at or above the threshold are “false positives” (FP), coding sequences below a threshold are “false negatives” (FN), and negative control sequences below a threshold are “true negatives” (TN). An optimal score threshold is selected for each tool based on the minimal distance to the [0.0,1.0] coordinate on the ROC plot (i.e. “closest.topleft” in the `R pROC` package). This best balances the sensitivity and specificity for each tool [31].

For each possible score threshold the *sensitivity*, *specificity* and *Matthews correlation coefficient (MCC)* are computed, these are defined below. Furthermore, a “receiver-operator characteristic” (ROC) plot is also generated for each tool (Figure 3A) using the pROC package (version 1.18.5) [31]. The “AUC” or Area Under the ROC curve is computed for each tool as a further performance measure, which we note has a theoretical range $[0.5, 1.0]$, and is robust to potential issues of class imbalance [32].

$$Sensitivity = \frac{TP}{TP+FN},\quad Specificity = \frac{TN}{TN+FP},\quad MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

The runtimes for each tool were collected on a standard, isolated desktop computer (Intel 12-core processor i9-8950HK, 32GB RAM) running a Linux operating system (Ubuntu 22.04.2 LTS, Linux kernel 6.2.0-36). We recorded times for predictions on 240 sequences with lengths ranging from 232 to 1114 nts. The three-frame, top-strand only tools were run on both the forward, and reverse complement sequences. The average runtimes for each tool can be viewed in Figure 3D, the dependency between sequence length and runtime can be seen in Supplementary Figure S2, average values can be viewed in Supplementary Table S4, column O.

Name	Input	Frames	Installation	Ease of Use	Outputs	Version	Reference
Single-sequence tools							
Bioseq2seq	Sequence	3	🙂	😐	🙁	2024-02-08	[33]
CPC2	Sequence	6	🙁	😐	🙂	2.0Beta	[34]
CPPred	Sequence	3	😐	🙁	🙁	2018-05-17	[35]
LGC	Sequence	3	😐	🙁	🙁	1.0	[36]
PLEK	Sequence	3	🙂	🙂	🙁	1.2	[37]
RNAsamba	Sequence	3	🙂	🙂	😐	0.2.4	[38]
tcode	Sequence	6	😐	🙂	😐	EMBOSS:6.6	[39, 40]
Comparative tools							
PhyloCSF	Alignment	6	🙁	😐	😐	1.0.1	[41]
RNAcode	Alignment	6	🙁	🙂	🙂	0.3	[42]

Table 1. Summary of assessed tools. We have annotated the type of input each tool requires, the number of frames they scan (frames 1-3 [fwd], or frames 1-6 [fwd+rev]), the version and references. In addition, tools have been assessed for their ease of installation on a modern *nix environment, and for their ease of use, and whether the outputs included useful information like ORF coordinates as well as scores. The categories for each were: Excellent: 🙂, Acceptable: 😐, Unsatisfactory: 🙁. Further details on these assessments can be found in **Supplementary Text**.

Tool inclusion criteria: In order to select tools for inclusion in the benchmark we iteratively searched the published literature (GoogleScholar, PubMed), examining previous benchmarks, software tool papers and reviews of the topic [1, 5, 19, 20]. We extracted candidate tools from the text and from the reference sections, these have been assessed against the following inclusion criteria: 1. The primary purpose of the tool is to predict protein-coding potential from nucleotide sequences or alignments, 2. The tool is publicly accessible, 3. The tool is able to be installed and is executed, 4. The tool is generalised, and not restricted to a single species or phylogenetic group, 5. The tool is unique, i.e. is not a clone or a slight modification of an existing tool, 6. The tool is **not** based on homology to known proteins, 7. The tool has been recently used, and there is evidence that it is or may be popular.

The tools that we assessed against these criteria are listed in Supplementary Table S1. A total of 36 tools were assessed and **75% (27)** did not meet at least one of the above criteria. The most common reasons for excluding a tool was the installation was unnecessarily complex or failed on a modern linux computing environment (criterion #3, 41% failed, 11/27), the tool was not generalised (criterion #4, 33%, 9/27) or was not publicly accessible (criterion #2, 26%, 9/27) (Figure 2A, Supplementary Table S1).

Brief tool descriptions

In the following section we briefly describe each of the software tools that met the above inclusion criteria for this study. See Supplementary Information for detailed information on sources of training and test data, the number of sequences used as positive and negative controls for each, whether summary statistics for each (e.g. mean length and C+G content) are reported and whether similarity between training and test datasets were accounted for.

Base-line tools

The authors of this manuscript added a naive coding potential finder “`stopFree`”, and employed a random number generator “`randScore`”. Both tools serve as base-line scores for a minimal and lowest expected performance in our evaluation that are free of any machine-learning based training. The simplest coding potential measure we have identified is the length of the

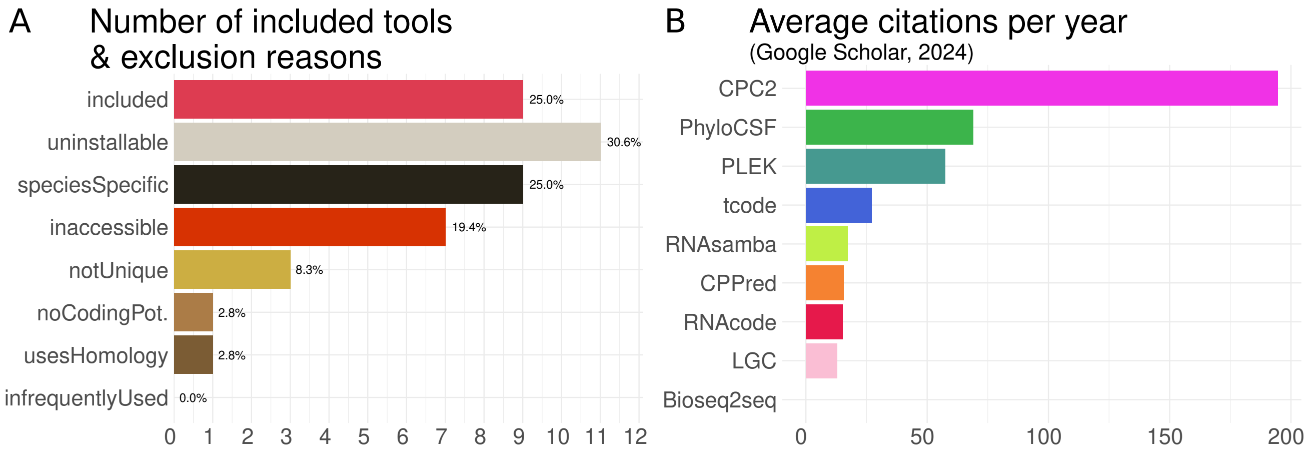


Figure 2. Software inclusion, exclusion and popularity. A. The proportions and counts of candidate software tools (total of 36), and the reasons some tools were excluded. See text and Supplementary Table S1 for further details. **B.** The average number of citations per year for each of the included tools (citation counts were collected in March 2024 from Google Scholar).

longest in-frame stop-free region for any input sequence (i.e. the longest run of UAA, UAG, and UGA free sequences over each of the six possible reading frames). This value conforms with the most general definition of an ORF [43].

We expect the stopFree measure to be robust to common sequence errors and non-canonical features such as truncations, non-AUG start codons [44], mis-splicing, and many forms of sequence error [45, 46]. We anticipate that stopFree as a minimal, single-metric tool will serve as a lower bound on tool accuracy for more sophisticated tools.

Meanwhile, randScore uses a random number generator that generates integers in the range $[1,100]$ for any input sequence, and is expected to be the worst possible prediction tool (also known as "monkey with a pencil").

Protein coding potential tools

Various approaches were employed by the tools that were benchmarked. We briefly describe them here, while more detailed tool descriptions, discussion of their training and test datasets, sequence/alignment features, command-line parameters, installation, user experience and output notes can be found in the Supplementary Information.

For the sequence-based tools, Bioseq2seq [33] and RNAsamba [38] use neural network models. CPC2 "Coding Potential Calculator 2" [34], CPPred "Coding Potential PREDiction" [35] and PLEK [37] utilise support vector machines. LGC "ORF Length and GC content" [36] employs a maximum likelihood method. tcode [39, 40] uses an implementation of the Fickett TESTCODE statistic which is a probabilistic method.

For the alignment-based tools, PhyloCSF "Phylogenetic Codon Substitution Frequencies" [41] is a maximum likelihood method, while RNAcode [42] is a comparative analysis tool.

Author engagement

In order to ensure methods are used correctly, a preprint of these results and methods was shared with corresponding authors for each tool. We have elected to do this in order to minimise disagreements between benchmark and method authors, and ensure that we have adequately described, and not misrepresented any tools.

Results

Tool accuracy: Some clear trends have emerged from our analysis of coding sequence detection of reference datasets. The alignment-based tools RNAcode and PhyloCSF show a clear accuracy advantage over the single sequence approaches (Figure 3A-C). This ranking holds across each accuracy measure (Area Under ROC Curve (AUC), Sensitivity, Specificity and the Matthews Correlation Coefficient (MCC)) with RNAcode consistently outperforming PhyloCSF. The tcode tool, which is an implementation of a method proposed in 1982 is not only faster, but also significantly more accurate than modern sequence-based tools.

We were particularly surprised to find that a naive tool, stopFree, that simply reports the length of the longest stop-codon-free region for a sequence, outperforms the widely used tools CPC2, PLEK, LGC and CPPred. This was a consistent finding across the performance metrics and datasets. To assess this further, we performed a permutation-based test to compare the ROC curves for each tool against the baseline stopFree tool (Figure 3B) [47]. Three tools performed significantly better than the baseline stopFree tool: RNAcode (z score$= 33.2$, $p = 4.3x10^{-241}$), PhyloCSF (z score$= 23.1$ $p = 1.3x10^{-117}$), and 1982 sequence-based tool tcode (z score$= 6.1$, $p = 1.01x10^{-9}$). The performances of bioseq2seq and CPC2 were not significantly better or worse than the stopFree tool. Whereas RNAsamba, CPPred, PLEK and LGC produced ROC curves that indicated a significantly worse performance than the stopFree baseline ($p < 0.05$ for each).

Self-assessment disparities: For most tools we note that there were large differences between the self-reported accuracies and the ones we calculated. While this trend has been reported previously [48], we were surprised by the degree of discrepancy (black arrows, Figure 3B & Supplementary Figure S1). The RNAcode publication reports the lowest self-reported accuracy statistics of all included tools, and yet is the most consistent with our independent accuracy measurements (Figure 3A, Supplementary Figure S1, Supplementary Table S4). All self-reported accuracy statistics for the other tools have a much larger discrepancy with our results, which are ranked by degree of disparity in Supplementary Figure S1B. The possible causes of these discrepancies are discussed further in the conclusions.

Computational efficiency of selected tools: In terms of computational efficiency, we found that the linear, composition and sequence-based tools tcode, stopFree, LGC and PLEK shared roughly equivalent runtimes of $0.01 - 0.03$ seconds per sequence. CPPred, CPC2 and RNAcode had similar runtimes of $0.1 - 0.4$ seconds per sequence, and bioseq2seq, RNAsamba and PhyloCSF are slower tools, needing $2.0 - 50$ seconds per sequence (See Figure 3D, Supplementary Table S4). As can be seen in Supplementary Figure S2, most tools are relatively insensitive to sequence length at these length scales, with only the alignment-based methods showing a clear dependency between runtime and sequence length.

Integrity of the control data sets: For most of the tools we tested, the coding potential scores for the two negative control sets (intergenic and shuffled sequences) of each clade (Animalia, Fungi, and Plantae) are generally very consistent (Figures S6-S8), indicating that both shuffled and random intergenic regions can serve as valid negative controls for coding annotation tools, except two tools, RNAsamba and LGC.

The RNAsamba scores show a large differences between animal intergenic and shuffled sequences, with the shuffled scores being high and similar to the values for coding sequences. This suggests that RNAsamba identifies intergenic sequences rather than coding sequences when run on animal and plant derived sequences Figure S8.

The LGC scores for plant genomes have a large discrepancies between the intergenic and shuffled negative controls, which may be due to the low C+G content in plant intergenic regions. These regions received higher scores than coding sequences, as shown in Supplementary Figure S7. This suggests a flaw in the LGC model's ability to differentiate between non-coding and coding sequences with low C+G content [36].

Relationship between citations and accuracy and speed: As observed in previous studies, citations rates do not reflect the accuracy or computational efficiency of bioinformatic tools [49]. This lack of a relation can be confirmed here by comparing tool rankings from Figure 2B and Figure 3B In particular, RNAcode citations are approximately the same as two of the tools that performed barely better than a random number generator. In our hands RNAcode consistently outperforms competing tools. Meanwhile, the most highly cited, and presumably most widely used tool CPC2, performed no better than the baseline stopFree scores while being one of the least computationally efficient sequence based tools (Figures 2B & 3D).

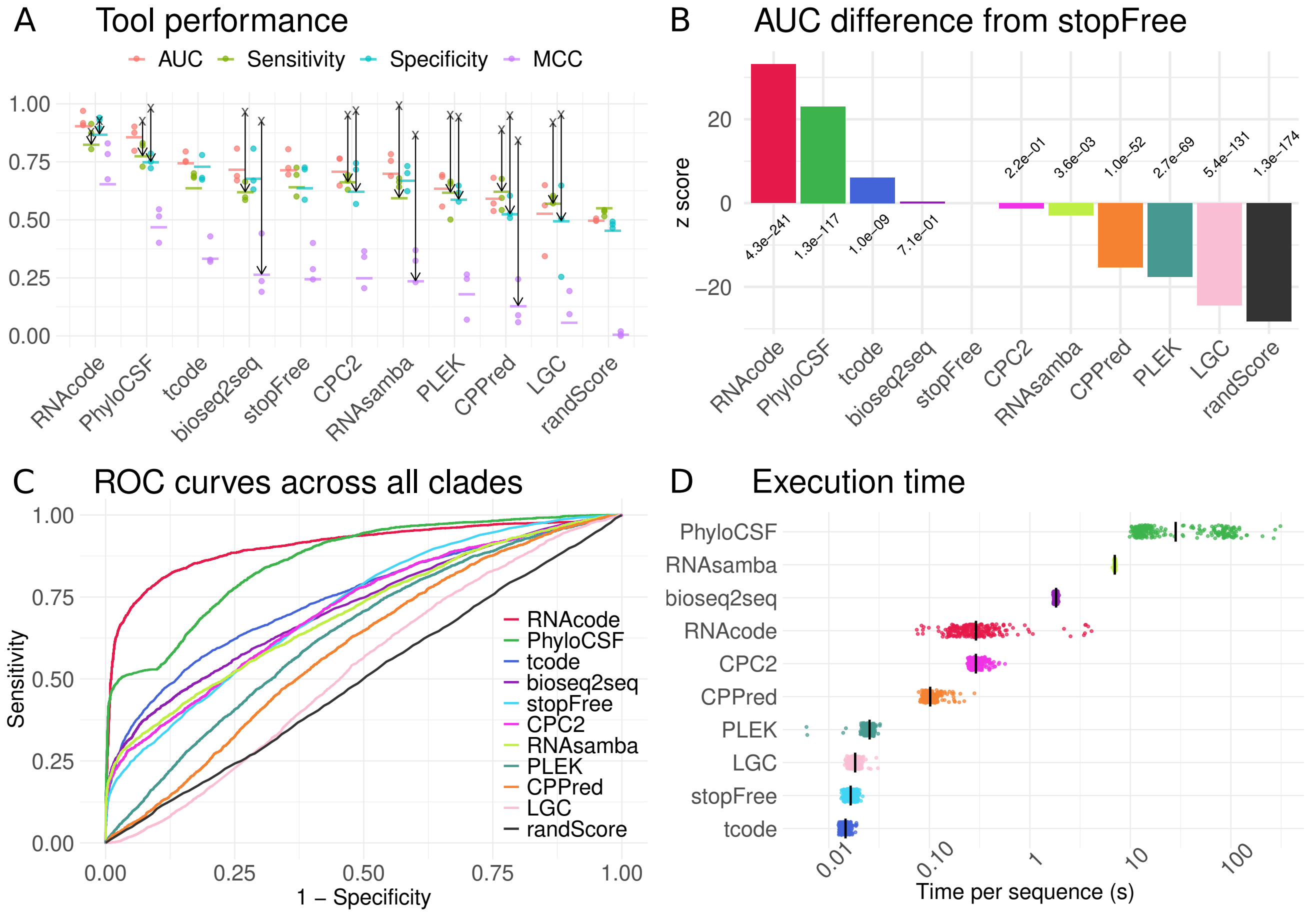


Figure 3. Coding potential tools, accuracy and speeds. A. A jitter plot showing the mean Area Under ROC Curve (AUC), Sensitivity, Specificity and Matthews Correlation Coefficient (MCC) values for each software tool and clade. Tools are ordered by the median AUC (mean performance values are indicated with a tick mark for each metric and tool). The black crosses and arrows indicate the accuracy metrics reported by the authors of each tool on their own test datasets. **B.** Bar plot showing Z-score differences of each tools ROC curves from our naive stopFree tool, with notated p-values. **C.** ROC curves showing the ability of each tool to discriminate between positive coding sequences and negative unannotated, length matched genome regions, and shuffled controls. The x-axis displays 1-Specificity (false-positive rate) and the y-axis displays the Sensitivity (true-positive rate). Each tool maps a trajectory from $(0.0, 0.0)$ to $(1.0, 1.0)$ through this plane as thresholds are lowered from the maximal to minimal value values for the tool on the test data. Ideal tools pass through the point $(0.0, 1.0)$. **D.** A jitter plot showing the runtimes for each tool, measured in seconds. Runtimes were recorded by running each tool on sequences of varying lengths on an isolated computer.

Conclusions

Evolutionary vs single sequence tools: Our evaluation shows that tools using evolutionary conservation patterns, like RNAcode and PhyloCSF, are more accurate than those based on single sequences alone. As we continue to expand the genomic tree of life [50, 51] , the usefulness of deploying such tools across different clades will increase significantly. These tools benefit from incorporating variations in nucleotide sequences that preserve protein functions, such as synonymous changes, conservative amino acid substitutions and frame-preserving insertions and deletions. This approach not only enhances accuracy but also supports the 'selected effect' definition of functional elements [52].

The only sequence based tool, tcode, to outperform the naive stopFree score is faster and more accurate that modern sequence-based tools. These three tools (RNAcode, PhyloCSF and tcode), published in 2011, 2011, and 1982 respectively, highlight how little genuine progress in the field has been made in recent years.

Challenges and recommendations for future development: This study adhered to key principles that should guide future tool development and evaluation. This is in contrast with some current tools that overlook essential aspects of mathematical modeling such as understanding the data and its applications. We highlight three main areas for consideration:

1. Sampling controls: We used well-annotated reference eukaryotic genomes outside any training dataset for our positive controls, questioning the need for species-specific tools. Given the conservation of the genetic code and translation machinery [53, 54], more generalized coding detection tools are feasible. Moreover, coding detection tools must be robust to biological and technical deviations from standard results, including truncated sequences, splicing errors, frame-shifting elements, recoding, non-AUG starts and non-standard genetic codes [45, 55, 56, 44, 46, 54]. *2. Dataset construction:* Both positive and negative datasets must be challenging and well-matched to control confounding factors such as sequence length and C+G content. This ensures that coding sequence scores will be robust and distinct from genomic background. Using short and long ncRNAs as negative controls is insufficient (see below). *3. Dataset representation:* Evaluation datasets should mirror the typical genomic composition, which for many species includes a predominance of junk and functional non-coding sequences. Evaluation metrics must be robust against imbalanced datasets [57].

In this study, several widely used tools performed poorly in distinguishing coding from non-coding sequences, illustrating once more that software popularity does not equate to accuracy or efficiency [49]. Despite high self-reported accuracy, our independent evaluations revealed large discrepancy with measurements, suggesting the need for more rigorous validation measures (See Figures 3A and S1) [48].

Stop using highly curated datasets for training and testing: The meticulously curated mRNA sequences from RefSeq and GENCODE [58, 59] are, in our view, unsuitable as positive controls for training or testing coding annotation tools. These datasets have undergone extensive refinement over decades by many curators, and have incorporated additional experimental data such as Ribo-seq and mass-spec to further support annotations [60, 61]. However, typical real-world datasets contain errors such as truncations, sequencing and assembly problems that can cause frame-shifts and mis-spliced transcripts (to list just a few discrepancy sources). Consequently, highly curated datasets are a trivial problem set that exaggerate the significance of inherent sequence features, such as the longest canonical ORFs. This is likely the cause of the many misclassifications by many tools in our practical assessment scenarios.

The problematic selection of negative training data sets is a further issue. We noted that either short or long ncRNAs have been used as negative datasets for training some of the above tools (see tools descriptions in the Supplementary Materials). Rfam-derived datasets are an inadequate negative control for this problem as these short non-coding RNAs have profoundly different lengths and sequence compositions compared to coding and typical genomic sequences. Furthermore, the majority of lncRNAs have been labelled lncRNAs because they have (a) some evidence of transcription, and (b) lack an obvious open reading frame [4]. Therefore, lncRNAs have shorter ORFs than random sequences, and therefore are a trivial and inadequate negative control for training or testing a coding-potential prediction tool. Finally, lncRNAs have a very different length distribution relative to mRNAs, therefore trivial tools could distinguish between these transcripts using measures as simple as sequence length.

In short, the tools that underperformed in this benchmark are designed to solve a nonexistent problem in genomics: distinguishing error-free, curated coding sequences from shorter non-coding RNA sequences.

Class balance: We noted that many tools trained and evaluated their performance on relative balanced classes of negative and positive datasets. In reality the ratio of coding to non-coding is likely to be low in eukaryotes (approximately 1 to 100 the human genome for example). In order to evaluate performance under realistic conditions we recommend that realistic positive to negative class ratios are maintained. Assessments should use metrics that are robust to class imbalance, rather than artificially attempt to balance positive and negative classes [57, 32].

Limit sequence similarity between test and training: a standard practice for applying machine learning techniques in computational biology is to ensure test and validation sets are freed of homologous or similar sequences found in the training set [62, 63, 64]. Some of the tools assessed here did not report controlling for evolutionary relationships between their training and test sets (See "tool descriptions" in the Supplementary Information), as a consequence may have artificially inflated the performance measures for their tool.

Study limitations: Firstly, this study is very eukaryote-specific, we have not assessed tools on more diverse genomic data from organelles, protists, archaeal, bacterial or viral sequences. Species that utilise novel genetic codes may prove to be a particularly challenging cohort for assessments [54].

Secondly, we have used a relatively balanced positive and negative datasets (ratios of approximately 1:3). On more representative datasets the negative datasets vastly outnumber the positive (e.g. approximately 98.5% of the genome is non-coding in human). So, ideally aim for approximately 30-50 times more negative sequences as positive to reflect realistic coding:non-coding ratios [57].

Thirdly, we have “abused” some tools by not applying them to the scenarios for which they were developed. For example, transcriptome tools that expect positive stranded and full-length ORFs in sequences free of introns. Also, our intergenic negative control regions are likely to be less well conserved than coding sequence. However, the alignment depths and percent-identity are not terribly dissimilar (Table S3), whereas the shuffled alignments have identical depth and identity. We found there was little difference in score distributions between intergenic and shuffled for the alignment-based tool results, as a consequence believe the alignments for intergenic sequence to be valid negative controls.

Finally, we have not generated more complex sequence sets for evaluation. We did not attempt to assemble full length transcripts or simulate sequencing and assembly errors found in typical genome and transcriptome datasets.

Future efforts: We have highlighted the scope for further development of tools to address the coding sequence annotation challenge. We note that the existing tools have not employed some of the latest machine-learning methods, nor do they include more complex information for conserved peptide predictions, for example the increased covariation in coding sequence alignments has been observed in recent analyses [65, 66]. We also note there has not been much work in identifying the strengths of the statistical features that contribute to predictions.

From a biocuration perspective, there has been relatively little development of the “conservome” for coding sequences, i.e. large-scale screens of conserved sequence elements with characteristic signatures of evolutionarily conserved coding sequences [67]. These screens may prove very useful for classifying alternative splicing, alternate start or stop codons, short open reading frames and other conserved non-canonical protein coding sequences.

Finally, the results presented here further highlight the need for well maintained, robust, up-to-date and accurate software. To be achieved this will require further support for long-term software maintenance and career incentives that encourage these activities [68].

Availability

A github repository for the scripts, drafts, supplementary information, sequences, alignments and scores from each software tool is here:
https://github.com/Gardner-BinfLab/PCPBResults/

Acknowledgements

We have received valuable feedback from the Genomics Aotearoa research community.

Professor Mercelo Carazzolle and Dr. Antonio Camargo (RNAsamba) for feedback on their tool.

Funding

This research was supported by an MBIE smart idea PROP-61726-ENDSI-PFR (DJC, THC, ST, PPG), the MBIE data science platform “Beyond Prediction: explanatory and transparent data science” (DJC, MAB, PPG), Genomics Aotearoa (MAB, PPG), and Marsden Grants 19-UOO-040 and 20-UOA-282 (PPG).

References

[1] James W Fickett and Chang-Shung Tung. Assessment of protein coding measures. *Nucleic acids research*, 20(24):6441–6450, 1992.

[2] Paulo Amaral, Silvia Carbonell-Sala, Francisco M De La Vega, Tiago Faial, Adam Frankish, Thomas Gingeras, Roderic Guigo, Jennifer L Harrow, Artemis G Hatzigeorgiou, Rory Johnson, et al. The status of the human gene catalogue. *Nature*, 622(7981):41–47, 2023.

[3] Daniel R Zerbino, Adam Frankish, and Paul Flicek. Progress, challenges, and surprises in annotating the human genome. *Annual review of genomics and human genetics*, 21:55–79, 2020.

[4] Jinyuan Xu, Jing Bai, Xinxin Zhang, Yanling Lv, Yonghui Gong, Ling Liu, Hongying Zhao, Fulong Yu, Yanyan Ping, Guanxiong Zhang, et al. A comprehensive overview of lncRNA annotation resources. *Briefings in bioinformatics*, 18(2):236–249, 2017.

[5] Jing Li, Xuan Zhang, and Changning Liu. The computational approaches of lncRNA identification based on coding potential: status quo and challenges. *Computational and Structural Biotechnology Journal*, 18:3666–3677, 2020.

[6] Adam Frankish, Sílvia Carbonell-Sala, Mark Diekhans, Irwin Jungreis, Jane E Loveland, Jonathan M Mudge, Cristina Sisu, James C Wright, Carme Arnan, If Barnes, et al. GENCODE: reference annotation for the human and mouse genomes in 2023. *Nucleic acids research*, 51(D1):D942–D949, 2023.

[7] Aoife Larkin, Steven J Marygold, Giulia Antonazzo, Helen Attrill, Gilberto Dos Santos, Phani V Garapati, Joshua L Goodman, L Sian Gramates, Gillian Millburn, Victor B Strelets, et al. FlyBase: updates to the Drosophila melanogaster knowledge base. *Nucleic acids research*, 49(D1):D899–D907, 2021.

[8] Stacia R Engel, Edith D Wong, Robert S Nash, Suzi Aleksander, Micheal Alexander, Eric Douglass, Kalpana Karra, Stuart R Miyasato, Matt Simison, Marek S Skrzypek, et al. New data and collaborations at the Saccharomyces Genome Database: updated reference genome, alleles, and the Alliance of Genome Resources. *Genetics*, 220(4):iyab224, 2022.

[9] Klas Hatje, Stefanie Mühlhausen, Dominic Simm, and Martin Kollmar. The protein-coding human genome: Annotating high-hanging fruits. *BioEssays*, 41(11):1900066, 2019.

[10] Alexander F Palazzo and Eliza S Lee. Non-coding RNA: what is functional and what is junk? *Frontiers in genetics*, 6:2, 2015.

[11] Jonathan M Mudge, Jorge Ruiz-Orera, John R Prensner, Marie A Brunet, Ferriol Calvet, Irwin Jungreis, Jose Manuel Gonzalez, Michele Magrane, Thomas F Martinez, Jana Felicitas Schulz, et al. Standardized annotation of translated open reading frames. *Nature biotechnology*, 40(7):994–999, 2022.

[12] Yi Qiu Alexander F. Palazzo and Yoon Mo Kang. mRNA nuclear export: how mRNA identity features distinguish functional RNAs from junk transcripts. *RNA Biology*, 21(1):1–12, 2024. PMID: 38091265.

[13] Roderic Guigó, Paul Flicek, Josep F Abril, Alexandre Reymond, Julien Lagarde, France Denoeud, Stylianos Antonarakis, Michael Ashburner, Vladimir B Bajic, Ewan Birney, et al. EGASP: the human ENCODE genome annotation assessment project. *Genome biology*, 7:1–31, 2006.

[14] Lincoln Stein. Genome annotation: from sequence to biology. *Nature reviews genetics*, 2(7):493–503, 2001.

[15] Y Amy Tang, Klemens Pichler, Anja Füllgrabe, Jane Lomax, James Malone, Monica C Munoz-Torres, Drashtti V Vasant, Eleanor Williams, and Melissa Haendel. Ten quick tips for biocuration. *PLoS computational biology*, 15(5):e1006906, 2019.

[16] Holly R Pinkney, Brandon M Wright, and Sarah D Diermeier. The lncRNA toolkit: databases and in silico tools for lncRNA analysis. *Non-coding RNA*, 6(4):49, 2020.

[17] Christopher Klapproth, Rituparno Sen, Peter F Stadler, Sven Findeiß, and Jörg Fallmann. Common features in lncRNA annotation and classification: a survey. *Non-coding RNA*, 7(4):77, 2021.

[18] Kiran Dindhoria, Isha Monga, and Amarinder Singh Thind. Computational approaches and challenges for identification and annotation of non-coding RNAs using RNA-Seq. *Functional & Integrative Genomics*, 22(6):1105–1112, 2022.

[19] Hansi Zheng, Amlan Talukder, Xiaoman Li, and Haiyan Hu. A systematic evaluation of the computational tools for lncRNA identification. *Briefings in Bioinformatics*, 22(6):bbab285, 2021.

[20] Dalwinder Singh and Joy Roy. A large-scale benchmark study of tools for the classification of protein-coding and non-coding RNAs. *Nucleic Acids Research*, 50(21):12094–12111, 2022.

[21] Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (CASP)—Round XIV. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1607–1617, 2021.

[22] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[23] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[24] Christina Nießl, Moritz Herrmann, Chiara Wiedemann, Giuseppe Casalicchio, and Anne-Laure Boulesteix. Over-optimism in benchmark studies and the multiplicity of design and analysis options when interpreting their results. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(2):e1441, 2022.

[25] Lukas M Weber, Wouter Saelens, Robrecht Cannoodt, Charlotte Soneson, Alexander Hapfelmeier, Paul P Gardner, Anne-Laure Boulesteix, Yvan Saeys, and Mark D Robinson. Essential guidelines for computational method benchmarking. *Genome biology*, 20:1–12, 2019.

[26] Chris P Ponting and Wilfried Haerty. Genome-wide analysis of human long noncoding RNAs: a provocative review. *Annual review of genomics and human genetics*, 23:153–172, 2022.

[27] Felipe A Simão, Robert M Waterhouse, Panagiotis Ioannidis, Evgenia V Kriventseva, and Evgeny M Zdobnov. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, 31(19):3210–3212, 2015.

[28] Mosè Manni, Matthew R Berkeley, Mathieu Seppey, Felipe A Simão, and Evgeny M Zdobnov. BUSCO update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes. *Molecular biology and evolution*, 38(10):4647–4654, 2021.

[29] 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.

[30] 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.

[31] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC bioinformatics*, 12:1–8, 2011.

[32] Matthew McDermott, Lasse Hyldig Hansen, Haoran Zhang, Giovanni Angelotti, and Jack Gallifant. A Closer Look at AUROC and AUPRC under Class Imbalance. *arXiv preprint arXiv:2401.06091*, 2024.

[33] Joseph D Valencia and David A Hendrix. Improving deep models of protein-coding potential with a Fourier-transform architecture and machine translation task. *bioRxiv*, pages 2023–04, 2023.

[34] Yu-Jian Kang, De-Chang Yang, Lei Kong, Mei Hou, Yu-Qi Meng, Liping Wei, and Ge Gao. CPC2: a fast and accurate coding potential calculator based on sequence intrinsic features. *Nucleic acids research*, 45(W1):W12–W16, 2017.

[35] Xiaoxue Tong and Shiyong Liu. CPPred: coding potential prediction based on the global description of RNA sequence. *Nucleic acids research*, 47(8):e43–e43, 2019.

[36] Guangyu Wang, Hongyan Yin, Boyang Li, Chunlei Yu, Fan Wang, Xingjian Xu, Jiabao Cao, Yiming Bao, Liguo Wang, Amir A Abbasi, et al. Characterization and identification of long non-coding RNAs based on feature relationship. *Bioinformatics*, 35(17):2949–2956, 2019.

[37] Aimin Li, Junying Zhang, and Zhongyin Zhou. PLEK: a tool for predicting long non-coding RNAs and messenger RNAs based on an improved k-mer scheme. *BMC bioinformatics*, 15:1–10, 2014.

[38] Antonio P Camargo, Vsevolod Sourkov, Gonçalo A G Pereira, and Marcelo F Carazzolle. RNAsamba: neural network-based assessment of the protein-coding potential of RNA sequences. *NAR genomics and bioinformatics*, 2(1):lqz024, 2020.

[39] James W Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic acids research*, 10(17):5303–5318, 1982.

[40] Peter Rice, Ian Longden, and Alan Bleasby. EMBOSS: the European molecular biology open software suite. *Trends in genetics*, 16(6):276–277, 2000.

[41] Michael F Lin, Irwin Jungreis, and Manolis Kellis. PhyloCSF: a comparative genomics method to distinguish protein coding and non-coding regions. *Bioinformatics*, 27(13):i275–i282, 2011.

[42] Stefan Washietl, Sven Findeiß, Stephan A Müller, Stefan Kalkhof, Martin Von Bergen, Ivo L Hofacker, Peter F Stadler, and Nick Goldman. RNAcode: robust discrimination of coding and noncoding regions in comparative sequence data. *Rna*, 17(4):578–594, 2011.

[43] Patricia Sieber, Matthias Platzer, and Stefan Schuster. The definition of open reading frame revisited. *Trends in Genetics*, 34(3):167–170, 2018.

[44] Dmitry E Andreev, Gary Loughran, Alla D Fedorova, Maria S Mikhaylova, Ivan N Shatsky, and Pavel V Baranov. Non-AUG translation initiation in mammals. *Genome Biology*, 23(1):111, 2022.

[45] Tamara Steijger, Josep F Abril, Pär G Engström, Felix Kokocinski, Tim J Hubbard, Roderic Guigó, Jennifer Harrow, and Paul Bertone. Assessment of transcript reconstruction methods for RNA-seq. *Nature methods*, 10(12):1177–1184, 2013.

[46] Marina V Rodnina, Natalia Korniy, Mariia Klimova, Prajwal Karki, Bee-Zen Peng, Tamara Senyushkina, Riccardo Belardinelli, Cristina Maracci, Ingo Wohlgemuth, Ekaterina Samatova, et al. Translational recoding: canonical translation mechanisms reinterpreted. *Nucleic acids research*, 48(3):1056–1067, 2020.

[47] ES Venkatraman. A permutation test to compare receiver operating characteristic curves. *Biometrics*, 56(4):1134–1138, 2000.

[48] Stefan Buchka, Alexander Hapfelmeier, Paul P Gardner, Rory Wilson, and Anne-Laure Boulesteix. On the optimistic performance evaluation of newly introduced bioinformatic methods. *Genome biology*, 22(1):152, 2021.

[49] P P Gardner, J M Paterson, S McGimpsey, F Ashari-Ghomi, S U Umu, A Pawlik, A Gavryushkin, and M A Black. Sustained software development, not number of citations or journal choice, is indicative of accurate bioinformatic software. *Genome Biol*, 23(1):56, Feb 2022.

[50] Matthew J Christmas, Irene M Kaplow, Diane P Genereux, Michael X Dong, Graham M Hughes, Xue Li, Patrick F Sullivan, Allyson G Hindle, Gregory Andrews, Joel C Armstrong, et al. Evolutionary constraint and innovation across hundreds of placental mammals. *Science*, 380(6643):eabn3943, 2023.

[51] Lukas FK Kuderna, Jacob C Ulirsch, Sabrina Rashid, Mohamed Ameen, Laksshman Sundaram, Glenn Hickey, Anthony J Cox, Hong Gao, Arvind Kumar, Francois Aguet, et al. Identification of constrained sequence elements across 239 primate genomes. *Nature*, pages 1–8, 2023.

[52] Stefan Linquist, W Ford Doolittle, and Alexander F Palazzo. Getting clear about the f-word in genomics. *PLoS genetics*, 16(4):e1008702, 2020.

[53] Anton S Petrov, Burak Gulen, Ashlyn M Norris, Nicholas A Kovacs, Chad R Bernier, Kathryn A Lanier, George E Fox, Stephen C Harvey, Roger M Wartell, Nicholas V Hud, et al. History of the ribosome and the origin of translation. *Proceedings of the National Academy of Sciences*, 112(50):15396–15401, 2015.

[54] Yekaterina Shulgina and Sean R Eddy. A computational screen for alternative genetic codes in over 250,000 genomes. *Elife*, 10:e71402, 2021.

[55] Joseph K Pickrell, Athma A Pai, Yoav Gilad, and Jonathan K Pritchard. Noisy splicing drives mRNA isoform diversity in human cells. *PLoS genetics*, 6(12):e1001236, 2010.

[56] Heidi Dvinge and Robert K Bradley. Widespread intron retention diversifies most cancer transcriptomes. *Genome medicine*, 7:1–13, 2015.

[57] Sean Whalen, Jacob Schreiber, William S Noble, and Katherine S Pollard. Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics*, 23(3):169–181, 2022.

[58] Adam Frankish, Mark Diekhans, Irwin Jungreis, Julien Lagarde, Jane E Loveland, Jonathan M Mudge, Cristina Sisu, James C Wright, Joel Armstrong, If Barnes, et al. GENCODE 2021. *Nucleic acids research*, 49(D1):D916–D923, 2021.

[59] Bhanu Rajput, Kim D Pruitt, and Terence D Murphy. RefSeq curation and annotation of stop codon recoding in vertebrates. *Nucleic acids research*, 47(2):594–606, 2019.

[60] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):D733–D745, 2016.

[61] Eric W Sayers, Mark Cavanaugh, Karen Clark, Kim D Pruitt, Stephen T Sherry, Linda Yankie, and Ilene Karsch-Mizrachi. GenBank 2024 update. *Nucleic Acids Research*, 52(D1):D134–D137, 2024.

[62] Johannes Söding and Michael Remmert. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Current opinion in structural biology*, 21(3):404–411, 2011.

[63] Ian Walsh, Gianluca Pollastri, and Silvio CE Tosatto. Correct machine learning on protein sequences: a peer-reviewing perspective. *Briefings in bioinformatics*, 17(5):831–840, 2016.

[64] Samantha Petti and Sean R Eddy. Constructing benchmark test sets for biological sequence analysis using independent set algorithms. *PLOS Computational Biology*, 18(3):e1009492, 2022.

[65] Helena B Cooper and Paul P Gardner. Features of functional human genes. *bioRxiv*, pages 2020–10, 2020.

[66] William Gao, Ann Yang, and Elena Rivas. Thirteen dubious ways to detect conserved structural RNAs. *IUBMB life*, 75(6):471–492, 2023.

[67] Jonathan M Mudge, Irwin Jungreis, Toby Hunt, Jose Manuel Gonzalez, James C Wright, Mike Kay, Claire Davidson, Stephen Fitzgerald, Ruth Seal, Susan Tweedie, et al. Discovery of high-confidence human protein-coding genes and exons by whole-genome PhyloCSF helps elucidate 118 GWAS loci. *Genome Research*, 29(12):2073–2087, 2019.

[68] Adam Siepel. Challenges in funding and developing genomic software: roots and remedies. *Genome biology*, 20(1):147, 2019.

[69] Bi-Qing Li, Kai-Yan Feng, Lei Chen, Tao Huang, and Yu-Dong Cai. Prediction of protein-protein interaction sites by random forest algorithm with mrmr and ifs. *PLoS ONE*, 2012.

[70] Jack Kyte and Russell F Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, 157(1):105–132, 1982.

[71] Christopher Pockrandt, Martin Steinegger, and Steven L Salzberg. PhyloCSF++: a fast and user-friendly implementation of PhyloCSF with annotation tools. *Bioinformatics*, 38(5):1440–1442, 2022.

Supplementary information

Contents

S1.1 Detailed tool descriptions, command line parameters and installation, running and user-friendly status

In the following section we describe each of the software tools that met the inclusion criteria for this study. We have systematically identified where possible:

- the sources of training and test data
- the number of sequences used as positive and negative controls for each
- whether summary statistics for each (e.g. mean length and C+G content) are reported
- whether similarity between training and test datasets were accounted for
- self-reported accuracy statistics for each tool (sources are provided in Supplementary Table S4)
- whether or not all six frames are assessed by the tool and
- whether or not coordinates of predicted ORFs are reported.
- the command-line parameters
- justify the Excellent (☺), Acceptable (😐) or Unsatisfactory (☹) classifications for the installation, user experience and output of each tool (Table 1)
- Installation:
  - Is the tool well documented?
  - Is the tool current?
  - Are dependencies/other tool installations described?
  - Is package management used?
  - Where were the installation instructions? E.g. a text file, github, or other.
- User experience:
  - Does the tool have well-documented use cases?
  - Are tool options clearly documented?
  - Did the tool crash frequently?
  - Are there verbosity options that provide useful dialogue?
  - Were informative warnings printed?
  - Were examples provided?
- Output:
  - Was the output formatted in an easy-to-parse flat file (e.g. tsv, csv, gff, bed, . . .)
  - Was useful information provided (e.g. coding score/probability, ORF coordinates, ORF strand)
  - Was a complex wrapper required to screen the output?

S1.1.1 Base-line tools

stopFree and **randScore**: The authors of this manuscript constructed a naive coding potential finder, and employed a random number generator. Both tools serve as base-line scores for a minimal and lowest expected performance in our evaluation. The simplest coding potential measure we identified is the length of the longest in-frame stop-free region for any input sequence (i.e. the maximum run of UAA, UAG, and UGA free sequences over each of the six possible reading frames). This conforms with the most general definition of an ORF as discussed by Sieber *et al.* [43].

We expect the stopFree measure to be robust to common sequence errors and non-canonical features such as truncations, non-AUG start codons [44], mis-splicing, and many forms of sequence error [45, 46]. We also note that no machine learning methods and no training data were used in the development of this tool, and as far as we are aware this is the first assessment of this metric. We anticipate that stopFree as a minimal, single-metric tool that will serve as a lower bound on tool accuracy for more sophisticated tools.

Meanwhile, randScore uses a random number generator that uses the integers in the range $[1,100]$ for any input sequence, and is expected to be the worst possible prediction tool (also known as “monkey with a pencil”).

The command-line parameters used for this evaluation were:

```
$ python3 stopFree input.fa outputFile

$ randScore=$((1 + $RANDOM % 100)); echo $randScore  > outputFile
```

S1.1.2 Sequence-based tools

Bioseq2seq is a neural network model of nucleotide to protein translation [33]. It is unclear from the bioseq2seq paper what and how many specific variables are extracted from each sequence, but we understand that a Fourier transform is used to capture potential sequence periodicity signals. The training data is derived from RefSeq (release 200) transcript and protein sequences derived from eight mammalian species (*Bos taurus*, *Pan troglodytes*, *Gorilla gorilla*, *H. sapiens*, *Macaca mulatta*, *M. musculus*, *Pongo abelii*, *Rattus rattus*). From 63,272 lncRNA and mRNA transcripts $< 1,200$ nts training, validation and test sets were sampled, using an 80/10/10 proportion for each. The similarity between the sets was restricted using CD-HIT-EST-2D, BLAST and Needleman-Wunsch alignment to exclude transcripts $> 80\%$ nucleotide sequence similarity. The test set contained 1,913 lncRNAs and 1,791 mRNAs, approximately a 1:1 ratio. The length and C+G contents for both the positive and negative datasets is not mentioned. The self-reported overall accuracy statistics for bioseq2seq are an F1 of 0.961, a sensitivity (recall) of 0.963, a PPV (precision) of 0.960, an MCC of 0.925 and AUPRC of 0.994 [33]. We further note that only the forward strand (frames 1, 2 and 3) are evaluated by this tool, and the coordinates of predicted ORFs are not provided in the output.

The command-line parameters used for this evaluation were:

```
$ python translate.py --checkpoint best_bioseq2seq-wt_LFN_mammalian_200-1200.pt
                      --input input.fa         --output outputFile
$ python translate.py --checkpoint best_bioseq2seq-wt_LFN_mammalian_200-1200.pt
                      --input input-REVCOMP.fa --output outputFile
```

Documentation for the install process located on github is simple and effective. Installation requires creating a Python virtual environment, using ‘pip’ with a requirements file and executing a bash install script. This ensures all dependencies are correctly versioned and not in conflict with installed packages. Installation is excellent. ☺

Usage documentation is clear and includes examples, making it easy to get started. However, it falls short in explaining the function of all parameters. Additionally, there are no simple warnings for user errors; it only throws general Python errors for missing or incorrect parameters. User experience is acceptable. 😐

Plain text output files are not formatted in an easy to parse manner. Metrics are distributed over multiple lines and have inconsistent spacing and colons as delimiters. There are only two useful metrics provided, coding probability and coding verdict. Output is unsatisfactory. ☹

CPC2 “Coding Potential Calculator 2” is an extension of CPC1 [34]. A random forest was used to select informative intrinsic features coding features (e.g. Fickett score [1], canonical ORF length, “ORF integrity” and isoelectric point). These features were used to train a support vector machine on approximately 18,000 high confidence *Homo sapiens* coding transcripts and 10,000 non-coding transcripts. The source of these transcripts is unclear from the manuscript since the numbers do not match the cited data source [34]. An “independent” test set of mRNAs was generated by intersecting the well curated Swiss-Prot and Refseq databases from *Homo sapiens*, *Mus musculus*, *Danio rerio*, *Drosphila melanogaster*, *Caenorhabditis elegans* and *Arabidopsis thaliana*. Non-coding transcripts were sourced from Ensembl and EnsemblPlants. Ratios between the positive and negative datasets ranged from approximately 1:2 to 4:1 across the species. Sequence similarity between the coding sequences was reduced to 90% sequence identity threshold with CD-hit. It is unclear from the methods section whether sequences homologous to the training dataset where removed from the independent test set. The self-reported overall accuracy statistic for CPC2 was 0.96, with specificity 0.97 and sensitivity of 0.95 [34]. Characteristic summary statistics for the positive and negative datasets such as length and C+G content were not reported. We further note that both the forward strand (frames 1, 2 and 3) and reverse (frames 4, 5 and 6) are evaluated by this tool.

The command-line parameters used for this evaluation were:

```
$ python2.7 CPC2.py  -r --ORF -i input.fa -o outputFile
```

Installation instructions available on their website are slightly confusing. The website download page details two options for installing. Method one is called “standalone”, which contains the filename “beta” and is based on python2.7. Method two which supports Python3 but has the filename “standalone”. The rest of the instructions pertain to the “standalone” version, which is seemingly for the python2.7 version “beta” and not the python3 version “standalone”. Biopython is required, but no version is stated. Installation is unsatisfactory. ☹

Documentation for running the tool consists of a single example run. The only options documented are found by running the tool without parameters. Incorrect usage results in useful error messages. User experience is acceptable. 😐

Output is provided in simple plain text and tab delimited format with headers and is easy to read and parse. Output includes important metrics such as coding probability, ORF start and Fickett score, as well as transcript and peptide lengths. Output is excellent. ☺

CPPred “Coding Potential PREDiction” is a Support Vector Machine (SVM) based coding prediction tool [35]. Thirty eight sequence features were ranked and selected using “mRMR-IFS” [69], these included the features used by CPC2 above, as well as hexamer scores, Gravy and protein instability metrics [70], and 30 “CTD” (composition-transition-distribution) features which appear to be mono- and di- nucleotide features, together with 20 “D descriptors” the definitions of which were unclear from the manuscript. A *H. sapiens* specific and an “integrated” model were trained on either *H. sapiens* alone, or *D. rerio*, *D. melanogaster*, *S. cerevisiae*, *C. elegans* and *A. thaliana* data. Positive coding sequences were obtained from NCBI RefSeq, while negative sets of sequences were based on ncRNAs from ENSEMBL release 90. We sampled the *H. sapiens* training datasets and found that the coding sequences had an average length of $\sim 3,800$ nts, while the average ncRNA length was ~ 800 nts. The C+G content of both sets is $\sim 47\%$. In total $\sim 50,000$ coding and $\sim 36,000$ ncRNA sequences were collected ($\sim 5:4$ ratio). These were randomnly assigned to either a training set (two thirds) or a test set (one third). Sequence similarity between the test and training sets was reduced to a 80% sequence identity threshold with CD-hit. The self-reported performance metrics of the integrated model on integrated test data was 0.92 MCC, 0.99 AUC, 0.96 accuracy, 0.97 sensitivity and 0.95 specificity [35]. We further note that only the forward strand (frames 1, 2 and 3) are evaluated by this tool, and the coordinates of predicted ORFs are not provided in the output.

The command-line parameters used for this evaluation were:

```
$ python2.7 CPPred.py  -i input.fa         -hex Integrated_Hexamer.tsv
                       -r Integrated.range -mol Integrated.model
                       -spe Integrated -o outputFile
$ python2.7 CPPred.py  -i input-REVCOMP.fa -hex Integrated_Hexamer.tsv
                       -r Integrated.range -mol Integrated.model
                       -spe Integrated -o outputFile
```

Installation instructions are available on their website. While the version of biopython is provided, the python2.7 dependency is not listed. Installation is acceptable. 😐

Usage documentation consists of providing a template example as well as multiple other examples with differing option variables. The program command can become overly complex as it requires six command line options to run. It is also not possible to run multiple instances at once due to hard coded temporary file names rather than randomised temporary filenames. The absence of error messages made troubleshooting difficult. User experience is unsatisfactory. ☹

The output files include over 40 data fields yet lack important information such as predicted start or stop sites and in which frame the provided score is found. The output file is tab-delimited, and easy to parse, however there is no documentation defining each output metric. Output is unsatisfactory. ☹

LGC “ORF Length and GC content” employs a maximum likelihood method to distinguish coding sequences from lncRNAs [36]. A polynomial is fit to the relationship between GC content and length of the top three ORFs (> 100 nts), parameters are estimated that minimise the root mean square error based on mRNA and lncRNA sequences from *H. sapiens*, *M. musculus*, *C. elegans*, *Oryza sativa* and *Solanum lycopersicum*. *H. sapiens* and *M. musculus* were obtained from RefSeq, lncRNAs from GENCODE. For the remaining species both sequence types were obtained from ENSEMBL, excluding ncRNAs < 200 nts. A total of $\sim 226,000$ protein-coding and $\sim 52,000$ lncRNA sequences were used in training (i.e. a ratio of coding positive to negative sequences of roughly 4 to 1). The summary statistics such as length and C+G content were not directly reported. The robustness of the approach is evaluated using all the plant, invertebrate, vertebrate and mammalian protein coding and ncRNAs from RefSeq (no version number was given, it is unclear if the mammal set is included within vertebrates too). Sequence similarity between the training and test sets was not discussed. The self-reported average performance metrics over six species for LGC were 0.94 accuracy, 0.92 sensitivity and 0.95 specificity [36]. We further note that only the forward strand (frames 1, 2 and 3) are evaluated by this tool, and the coordinates of predicted ORFs are not provided in the output.

The command-line parameters used for this evaluation were:

```
$ python2.7 LGC-1.0.py input.fa         outputFile
$ python2.7 LGC-1.0.py input-REVCOMP.fa outputFile
```

Installation documentation provided on their website is simple, but incomplete and out of date. Python 2.7 and biopython are required, however biopython now uses Python 3. The link to instructions on installing biopython is also no longer valid. The user needs to determine which version to install. Installation is acceptable. 😐

Usage documentation consists of providing one example. Command line options are non-existent with only an input and output being available. It is also not possible to run more than one instance of the software at a time due to non-random temporary file naming conventions. User experience is unsatisfactory. ☹

For output, documentation has simple descriptions of each of the output metrics. The output usefulness is hampered by not having any frame data or start and stop locations. The file is tab delimited and easy to collate data from, but unnecessarily repeats the documentation’s explanations of each of the output fields which increased the visual clutter of the output. While the majority of scores ranged between -10 and 10, sometimes an output of -10,000 is given. Output is unsatisfactory. ☹

PLEK “Predictor of long non-coding RNAs and messenger RNAs based on an improved k-mer scheme” uses a SVM to discriminate lncRNAs from protein-coding mRNAs [37]. PLEK uses 1,364 k-mer weighted frequency features for k values ranging from one to five (i.e. $\sum_{k=1}^{5} 4^k$ features). A SVM was trained on these features, with randomly selected 22,389 protein-coding transcripts from a *H. sapiens* RefSeq dataset (release 60) as a positive-set, and 22,389 *H. sapiens* long non-coding transcripts from GENCODE v17 formed a negative-set. In order to make the tool more robust to INDEL errors, INDELS of length zero to three were simulated with a rate ranging from 0% to 3%. A test set was generated by collecting novel transcriptome data for *H. sapiens* cell-lines MCF-7 and HeLa S3, the resulting sequences were allocated to either coding or non-coding sets by sequence alignment with RefSeq and GENCODE. There were 3,185 mRNAs and 121 lncRNAs corresponding to the MCF-7 data, and 3,045 mRNAs and 53 lncRNAs corresponding to the HeLa data. The potential for similarity between the training and test datasets was not discussed, and summary statistics such as length and C+G content were not reported, other than a limit $>$ 200 nt being used for each dataset. The self-reported average performance metrics over the two cell-line datasets were 0.947 and 0.955 sensitivity, and 0.958 and 0.925 specificity [37]. We further note that only the forward strand (frames 1, 2 and 3) are evaluated by this tool, and the coordinates of predicted ORFs are not provided in the output.

The command-line parameters used for this evaluation were:

```
$ python2.7 PLEK.py -fasta input.fa         -out  outputFile -thread #cpus
$ python2.7 PLEK.py -fasta input-REVCOMP.fa -out  outputFile -thread #cpus
```

Installation documentation on their website provides sufficient information to install and details prerequisites. Installation is excellent. ☺

Usage documentation is easy to follow and provides multiple examples as well as option parameter explanations. It works well with having multiple instances running and each can also be run with multiple cores using command line options. Erroneous usage produces useful error messages. User experience is excellent. ☺

Documentation provides no information on output files, however the output is a simple one-line text file. The only details in the output file are a coding verdict, followed by a score and the name of the sequence. The output usefulness is hampered by not having any frame or start and stop locations. The file is tab-delimited and easy to collate data from. Output is unsatisfactory. ☹

RNAsamba predicts the protein coding potential of RNA sequences using a neural network that incorporates variables from both RNA sequence and ORFs [38]. RNAsamba classifies input RNA sequences using canonical sequence features (ORF length, nucleotide k-mer frequencies, and amino acid frequencies) and a natural language processing neural network that uses the IGLOO architecture to identify informative motifs within the sequences without human input.

The training and test datasets employed are derived from multiple previously published datasets, these were CPC2 data (CCDS/RefSeq for mRNAs and GENCODE annotations of lncRNAs for a negative set from *H. sapiens*, *M. musculus*, *D. rerio*, *D. melanogaster*, *C. elegans* and *A. thaliana*). A FEELnc dataset from GENCODE (v24) *H. sapiens* transcripts with biotypes ’protein_coding’ or ’lincRNA’ & ’antisense’ for the negative set. A “mRNN” dataset derived from GENCODE (v25) contained a “regular” set, and a challenging set ($\leq$ 50 codon short ORFS, and $\geq$ 50 codon untranslated ORFs in a lncRNA set). The total number of sequences combining CPC2, FEELnc and mRNN datasets is 12,142 coding and 18,019 non-coding sequences. It was mentioned that transcript associated with loci present in test data were excluded from training data, but sequence similarity between these sets did not appear to have been controlled, other than for the truncated ORF dataset where MMseqs2 was used to remove sequences from the training set with $>$ 90% identity and coverage with the test set. Summary statistics such as length and C+G content were not reported for the positive and negative datasets. The reported average of performance over 5 species of RNAsamba a pre-trained model was 0.9915 sensitivity, 0.8891 PPV and 0.8639 MCC. We further note that only the forward strand (frames 1, 2 and 3) are evaluated by this tool, and the coordinates of predicted ORFs are not provided in the output.

```
$ rnasamba classify 'outputFile' 'input.fa'          'partial_length_weights.hdf5'
$ rnasamba classify 'outputFile' 'input-REVCOMP.fa' 'partial_length_weights.hdf5'
```

Installation documentation is available on their github code and io pages. Instructions are concise and allow for two routes to install, via pip or conda. Install options allow for installing a GPU version of the tool. Installation is excellent. ☺

Usage documentation contains instructions along with an example. Options were few but described. Error messages were useful. User experience is excellent. ☺

Output consists of a coding score and classification. Options allow to provide a protein fasta file output, however the output file does not contain frame information or any start and stop locations. Collating output data is simple as the output files contain little information and data is whitespace separated. Output is acceptable. 😐

tcode implements the Fickett TESTCODE statistic and is included in the EMBOSS suite of bioinformatic tools [39, 40]. This is a composition-based statistic that considers the frequency of each nucleotide in turn in each codon position. The statistical parameters for the model were estimated and evaluated on 321 coding and 249 non-coding viral, bacterial, *Homo sapiens* and *Saccharomyces cerevisiae* sequences that were available in 1982.

The command-line parameters used for this evaluation were:

```
$ tcode  -window 200 -outfile outputFile input.fa
```

Tcode is part of the EMBOSS suite. The documentation for installing EMBOSS is very detailed and complex and does not provide a simple install. We decided to install tcode using bioconda. Installation is acceptable. 😐

Usage documentation is extensive and covers general and specific use. Multiple options are clearly documented and examples are given. Error messages are useful. User experience is excellent. ☺

Documentation on output is extensive. The output is verbose, but does make it difficult to collate multiple runs. It provides sliding window scores and does not provide start and stop sites for coding regions precisely. Output is acceptable. 😐

S1.1.3 Alignment-based tools

PhyloCSF “Phylogenetic Codon Substitution Frequencies” is a maximum likelihood estimate for a given input alignment and phylogenetic tree using a 64x64 codon rate matrix trained on either coding or non-coding models [41]. Alignments are scored against both models using a log-likelihood ratio ($\log\left(\frac{p_c}{p_n}\right)$). PhyloCSF was recently reimplemented in C++, this new implementation “PhyloCSF++” is expected to be easier to install, and faster; However, it lacks features like an all-frame scanning option [71]. While PhyloCSF has modes that allow training on the specific dataset being studied, this option conflicts with our inclusion criteria. Therefore, we have used the generic “Omega Test” mode that requires a phylogenetic tree and not a species-specific empirical codon models that require the estimation of thousands of parameters. This mode is closely related to the widely used $\frac{d_N}{d_S}$ likelihood test. Based on the documentation, it is unclear how this model was parameterised, and whether positive and negative datasets were used in the development. While PhyloCSF performance measures were not reported directly in the manuscript, we can estimate these from ROC curve for “12-species alignments” and the “full dataset” shown in Figure 2A of the manuscript [41], the specificity is approximately 0.98 and the sensitivity is approximately 0.925. We further note that the coordinates of predicted ORFs are not provided in the output.

The command-line parameters used for this evaluation were:

```
$ PhyloCSF -f6 --removeRefGaps --strategy=omega modelFile input-alignment.fa
```

Installation instructions are substantial, but very complex. It requires installing a package manager for OCaml, then installation of multiple dependencies. It requires the user to compile PhyloCSF. We could not get OCaml to install correctly, which means we could not build PhyloCSF and install it using the provided instructions. We switched to using bioconda. Installation is unsatisfactory. ☹

Usage documentation is extensive, detailing user options, alignment preparation and provides multiple examples. A phylogenetic tree is required to use the “Omega” mode, which is a generalised coding score model. The tree and alignments must have exact naming and be formatted perfectly, and error messages did not help in identifying formatting as an issue. User experience is acceptable. 😐

Documentation on output is also extensive, detailing output metrics. The output contains the highest scoring region and corresponding strand, however it does not report start and stop coordinates. The strand start and alignment length are also provided. Extracting information from the output was easy due to simple tab-delimitted formatting. Output is acceptable. 😐

RNAcode [42]is a comparative analysis tool that identifies “evolutionary signatures” of protein coding sequences. A dynamic programming algorithm is used to find locally maximal scoring regions in an alignment with a reference sequence.

The score aggregates information from nucleotide and amino acid substitutions as well as frame-preserving gaps. The algorithm is tolerant of alignment and sequencing error so frame-shift due to errors are not over-penalised. The statistical significance of high scoring pairs is evaluated by simulating neutral alignments with similar evolutionary constraints. While machine-learning was not used to train this method, it was evaluated on enterobacterial, archaeal, insect, nematode and vertebrate coding and non-coding regions. The self-reported performance metrics for RNAcode were based on an automated annotation of Drosophila genome alignments and Flybase annotations were 0.88 sensitivity and 0.93 specificity.

The command-line parameters used for this evaluation were:

```
$ RNAcode -s -t input.aln -o outputFile
```

Documentation is extensive and provides options for root or non-root install as well as detailing multiple options when installing. Unfortunately we could not get it to install due to compile errors and switched to using bioconda for installation. Installation is unsatisfactory. ☹

Usage documentation is extensive, detailing user options, alignment preparation and provides multiple examples. Options were highly useful, as were error messages. User experience is excellent. ☺

Documentation on output is extensive and provides options on how the data is formatted. This made it very easy to collate multiple output files. Start and stop sites of the coding region were provided as well as strand. This was also one of the few tools to provide a p-value for the scores given. Output is excellent. ☺

S1.2 Accuracy and runtime figures

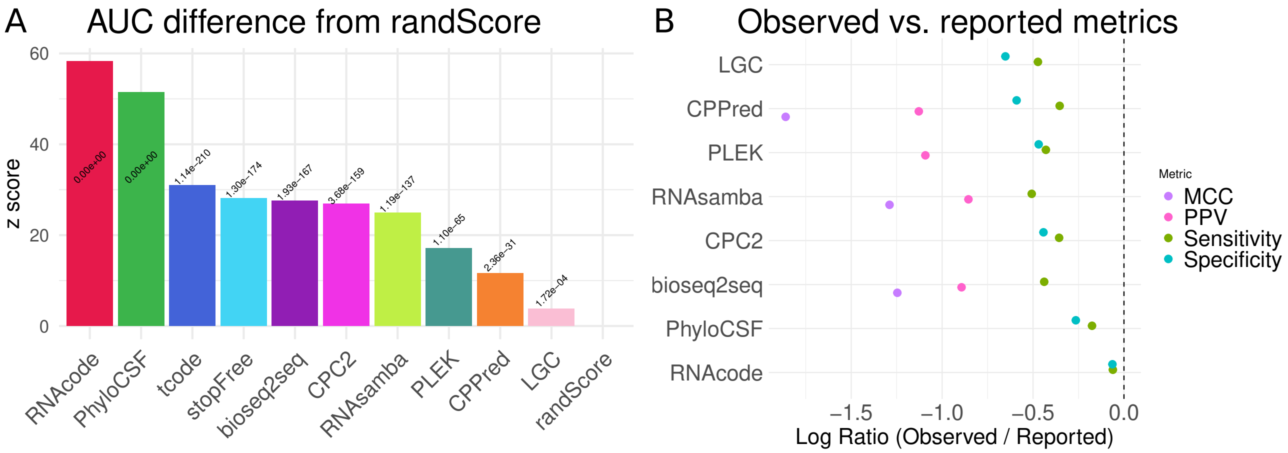


Figure S1. Tool accuracy. A z score differences in the area under the ROC curve for each tool compared to a random number generator, with corresponding p-values derived from bootstrap analysis. **B** Log ratios comparing self-reported to independently measured tool accuracy, where values near zero indicate high agreement and more negative values indicate greater discrepancies.

figures/Supp-TimeVLength.pdf

Figure S2. Time vs length scatter plot. The regression lines are shown for each tool. The initialisation times for each tool is given by the y intercept, the slope of the line shows the dependency of each tool on sequence length.

S1.3 Clade-specific result figures

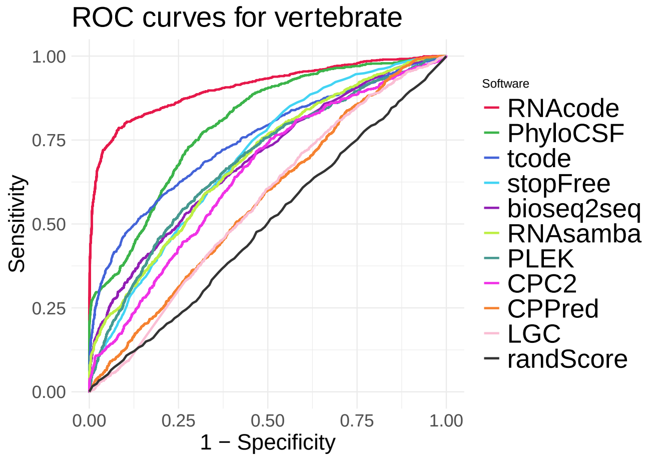


Figure S3. ROC curves for each tool with the vertebrate dataset.

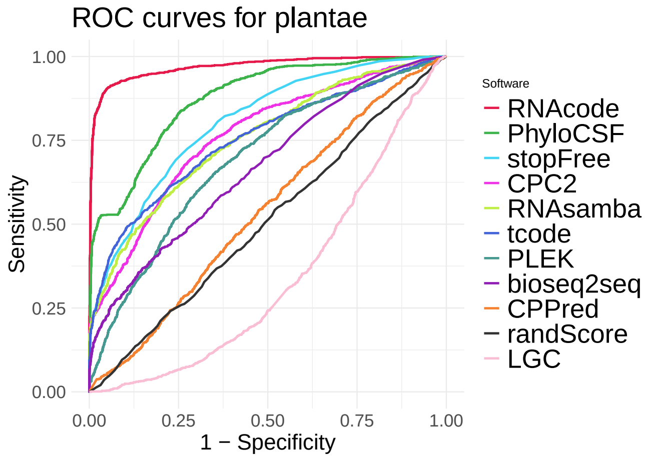


Figure S4. ROC curves for each tool with the plantae dataset.

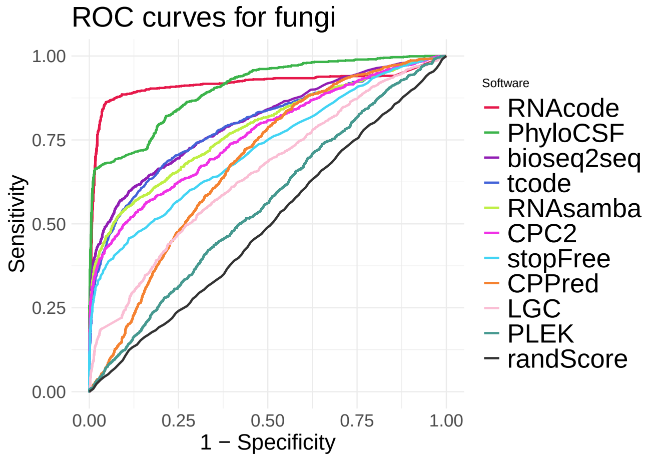
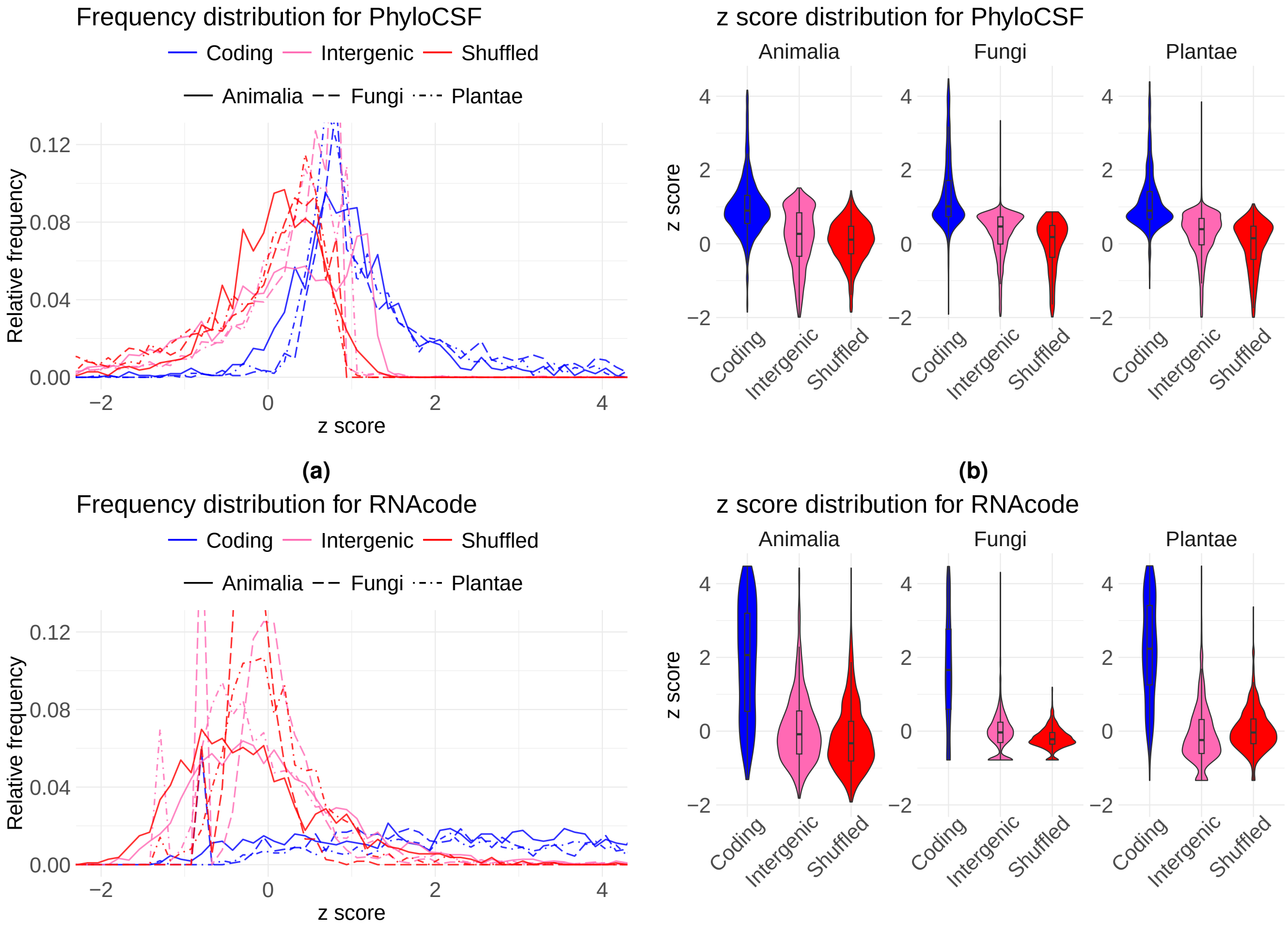


Figure S5. ROC curves for each tool with the fungi dataset.

S1.4 Score distributions figures

In order to compare score distributions $\tilde{s}$ for each tool we use a z score normalised for each tool. Within each clade a mean ($\bar{x}$) and standard deviation (σ) is computed from tool scores (x) for pooled negative controls of intergenic and shuffled sequences or alignments. To better cater for extreme distributions, the scores for bioseq2seq, CPC2 and CPPred were transformed with a $\log_{10}$ function.

$$z = \frac{x - \bar{x}}{\sigma}$$



(a)

(b)

(c)

(d)

Figure S6. Score distributions for alignment-based tools PhyloCSF (first row panels **a** & **b**) and RNAcode (second row panels **c** & **d**). The z scores for coding, intergenic, and shuffled alignments have been calculated for each test clade. The distributions are depicted as histogram line graphs (left panels **a** & **c**) and violin plots (right panels **b** & **d**) for each clade and data type.

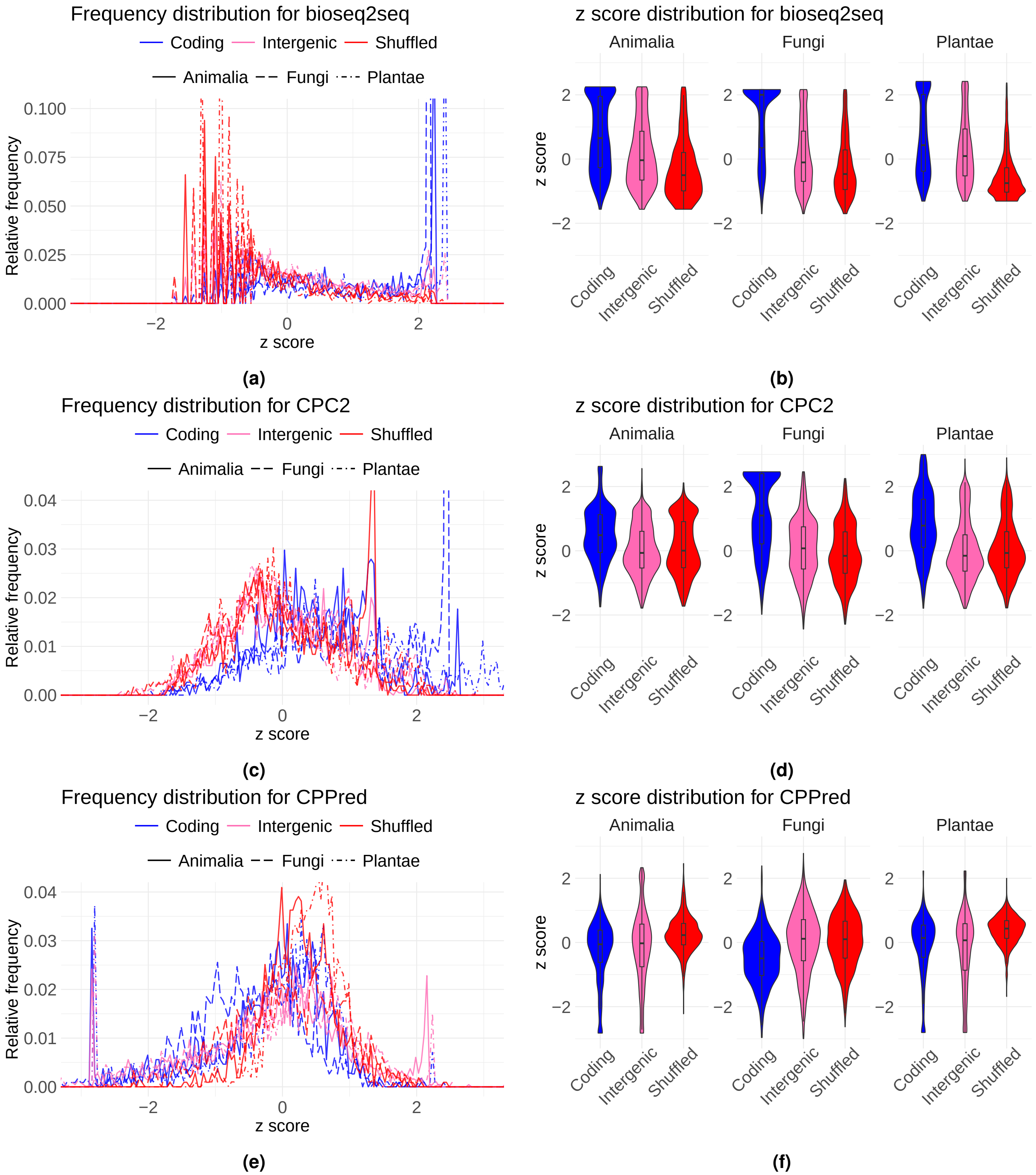
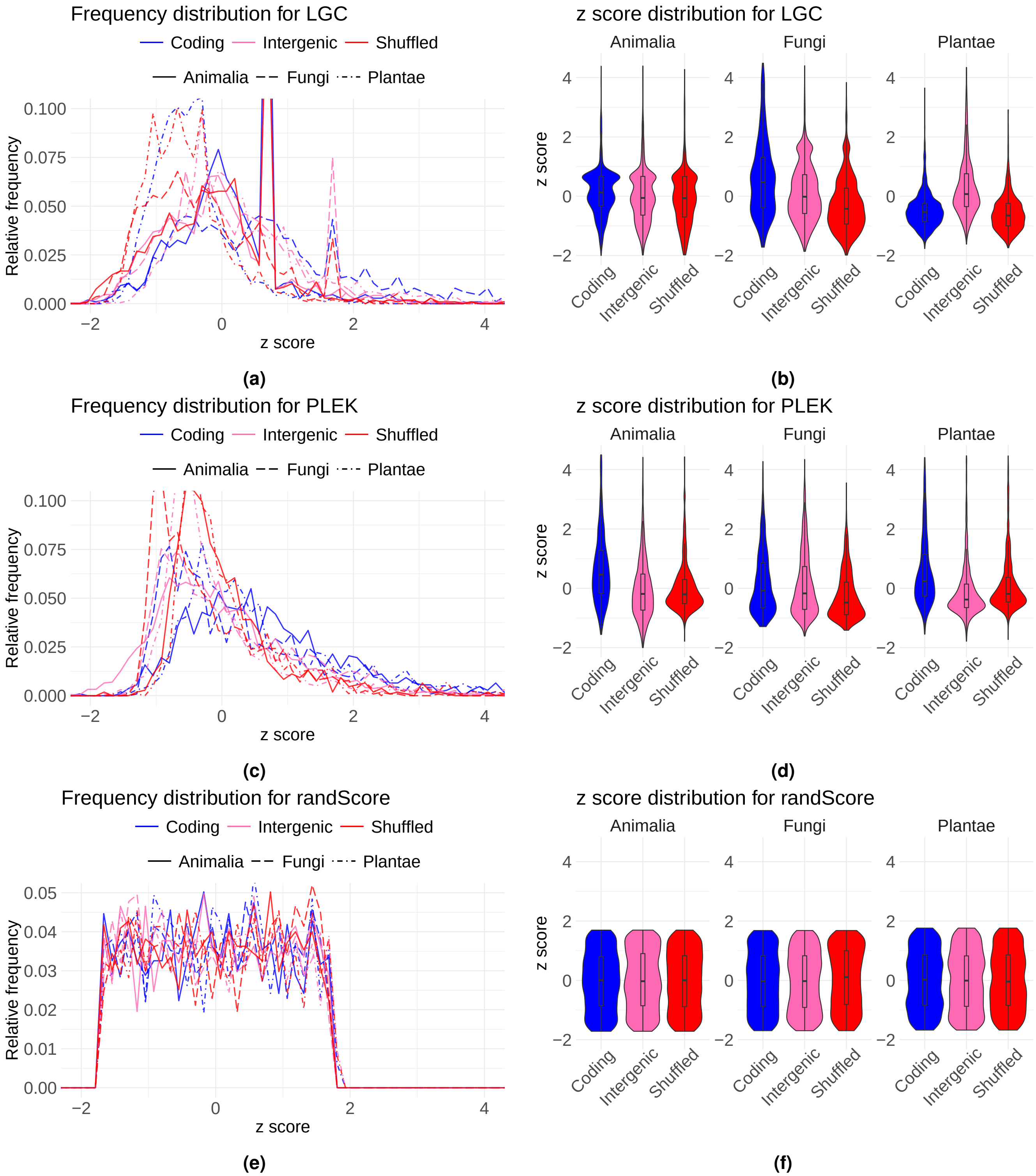


Figure S7. Score distributions for sequence-based tools bioseq2seq (first row panels **a** & **b**), CPC2 (second row panels **c** & **d**) and CPPred (third row panels **e** & **f**). The z scores for coding, intergenic, and shuffled sequences have been calculated for each test clade. The distributions are depicted as histogram line graphs (left panels **a**, **c** & **e**) and violin plots (right panels **b**, **d** & **f**) for each clade and data type.



(a)

(b)

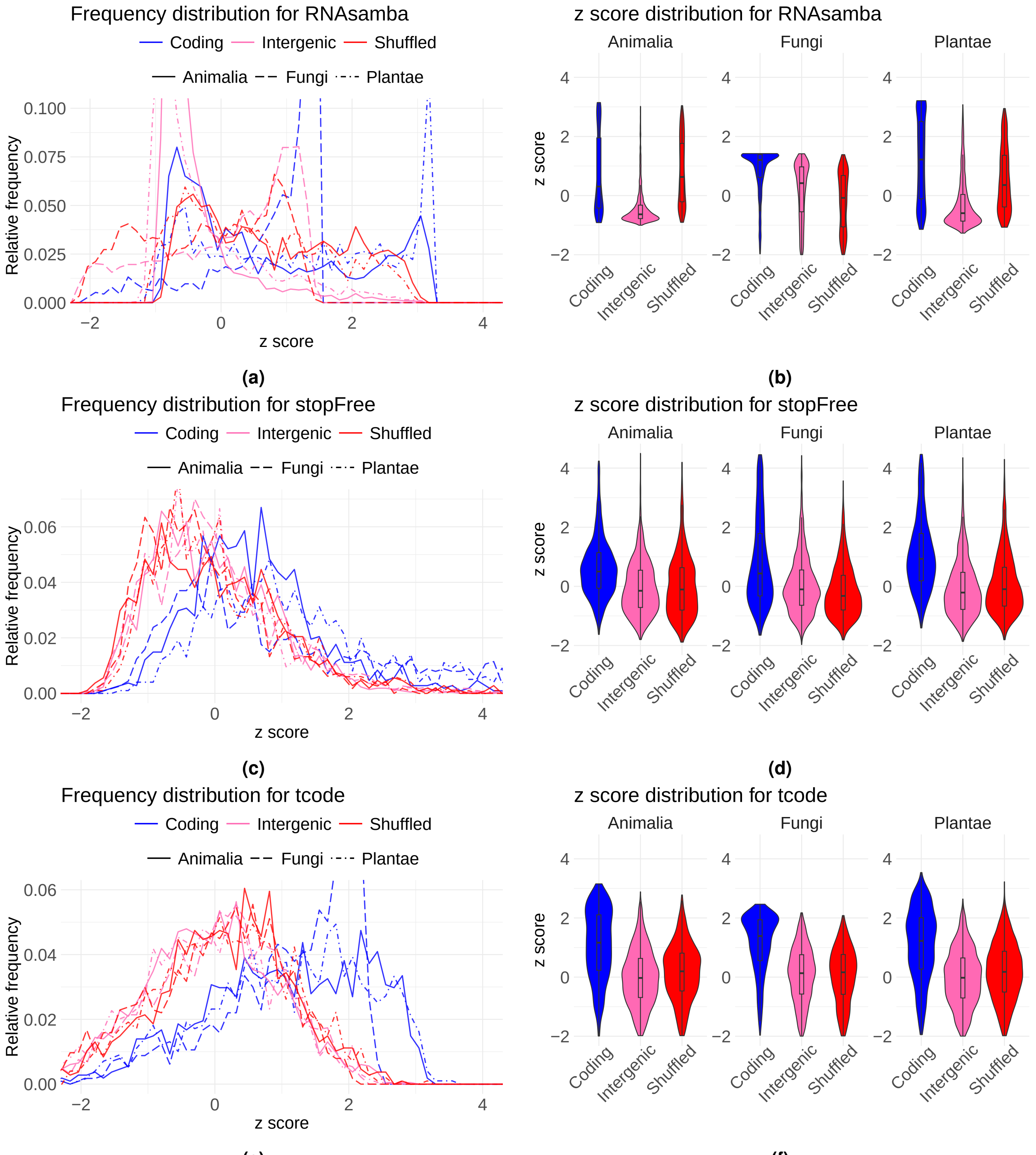
(c)

(d)

(e)

(f)

Figure S8. Score distributions for sequence-based tools LGC (first row panels **a** & **b**), PLEK (second row panels **c** & **d**) and randScore (third row panels **e** & **f**). The z scores for coding, intergenic, and shuffled sequences have been calculated for each test clade. The distributions are depicted as histogram line graphs (left panels **a**, **c** & **e**) and violin plots (right panels **b**, **d** & **f**) for each clade and data type.



(a)

(b)

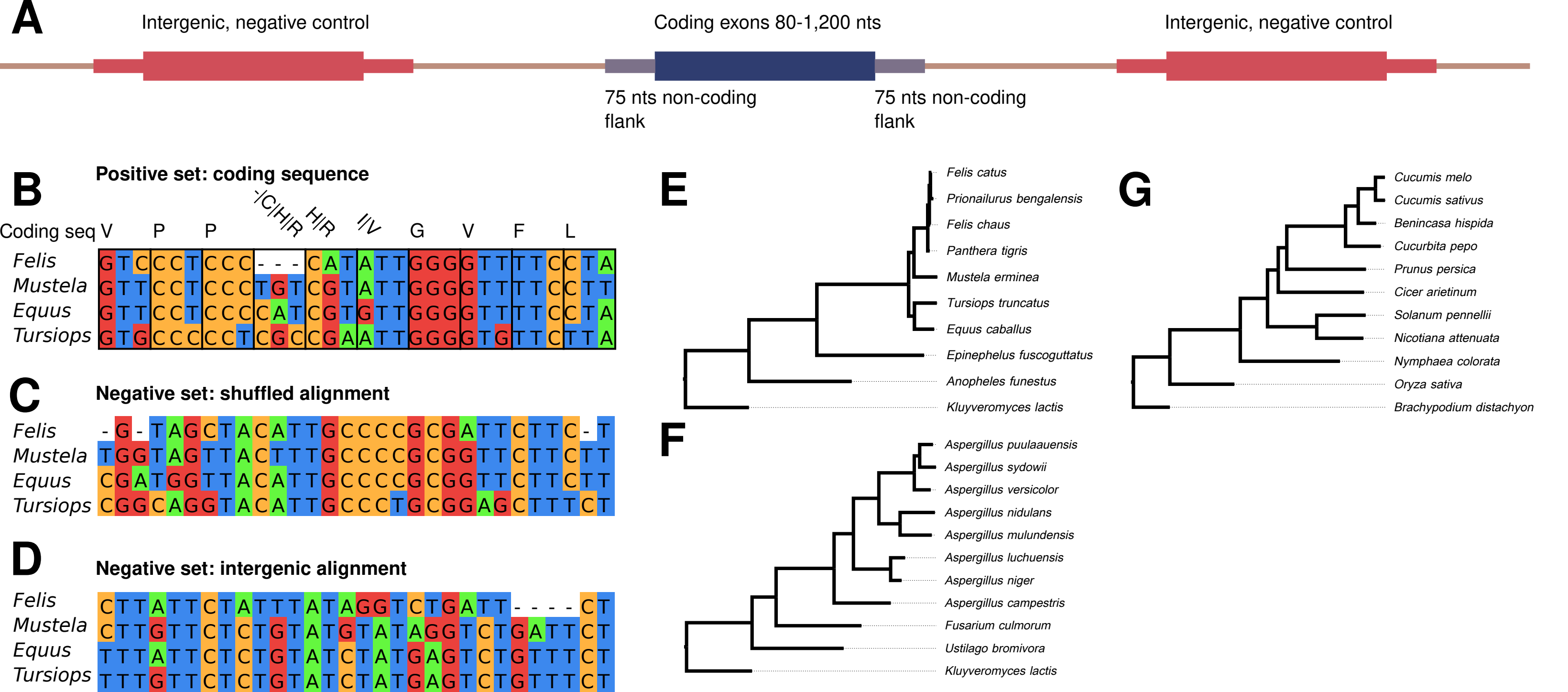
(c)

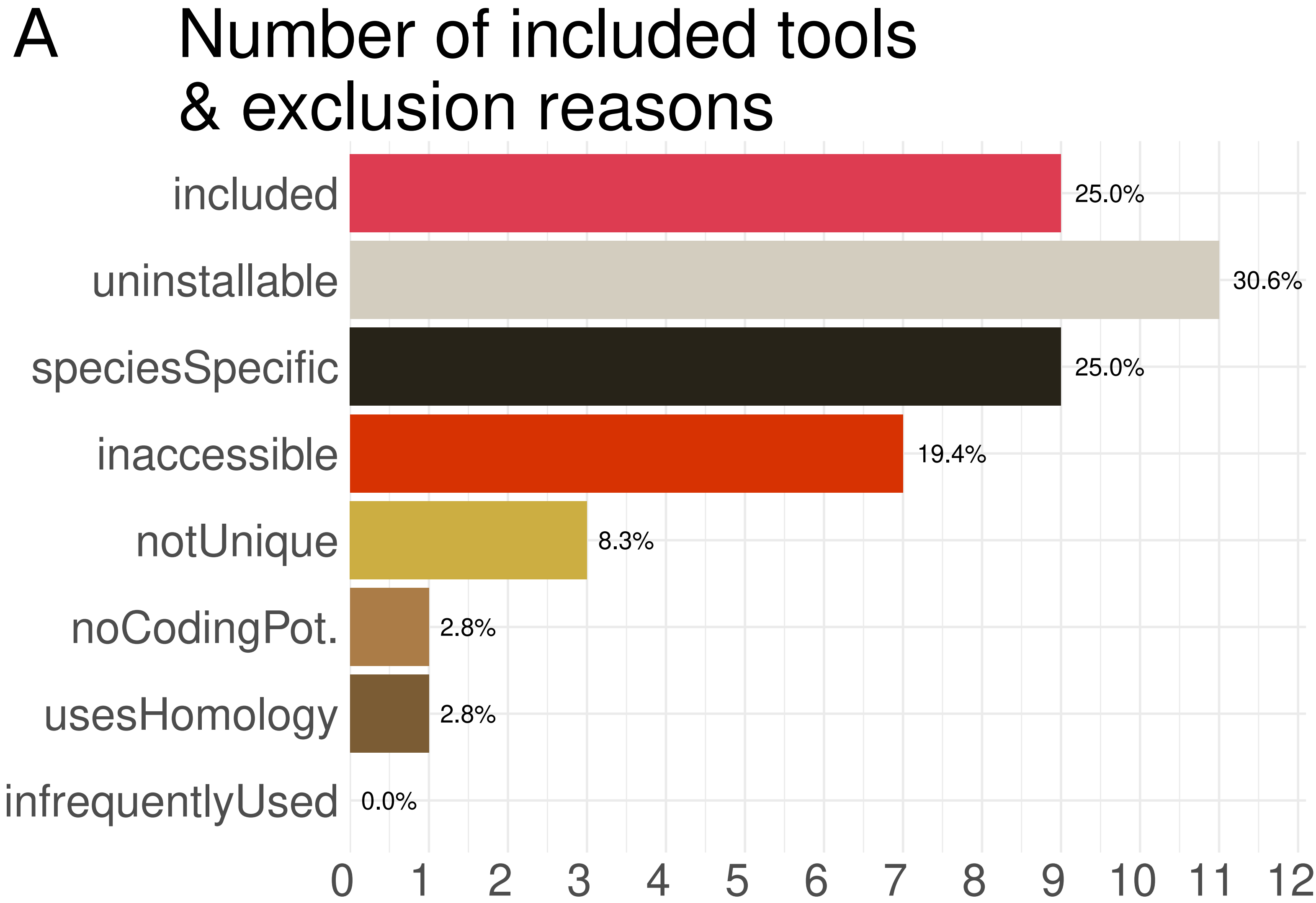
(d)

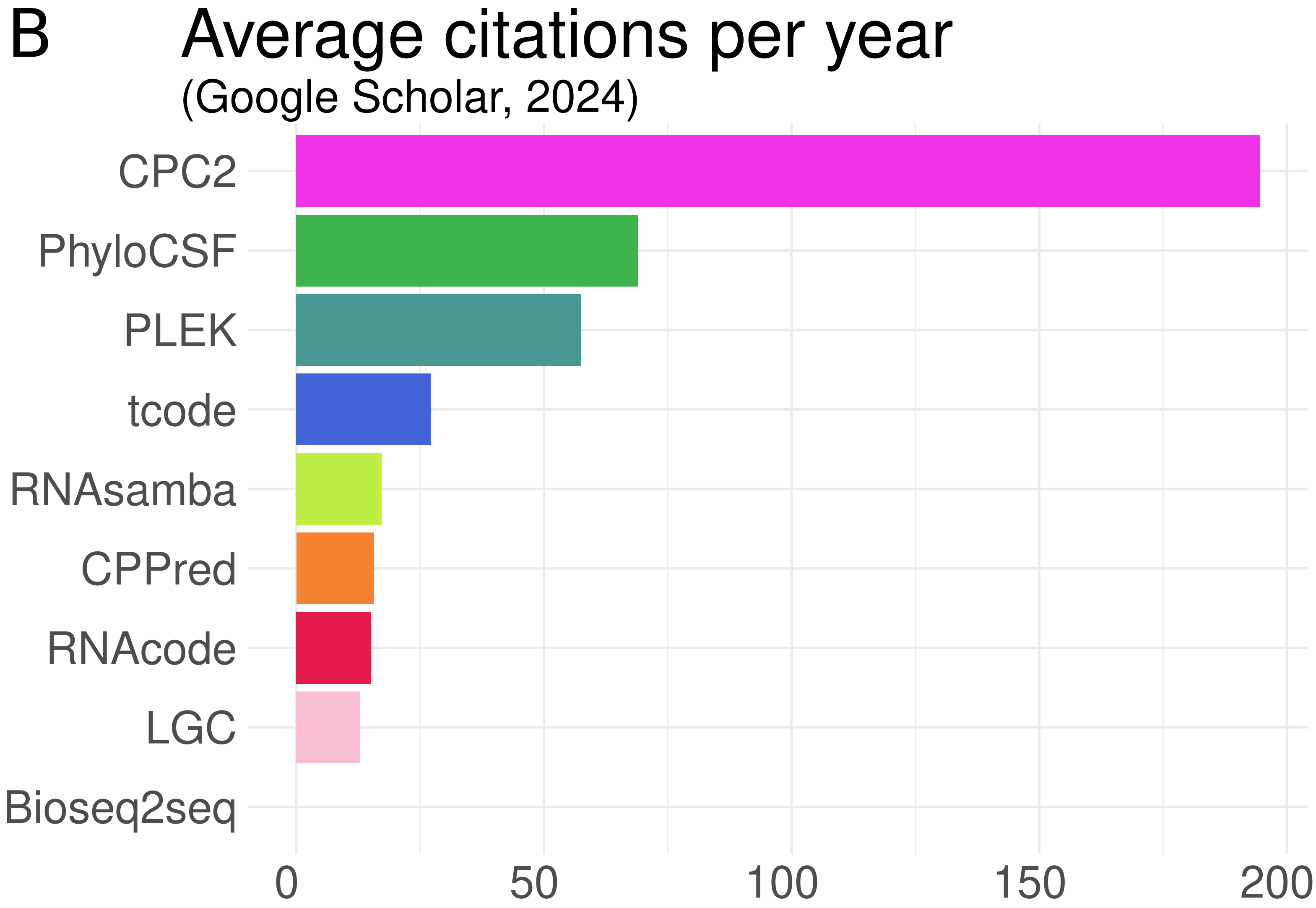
(e)

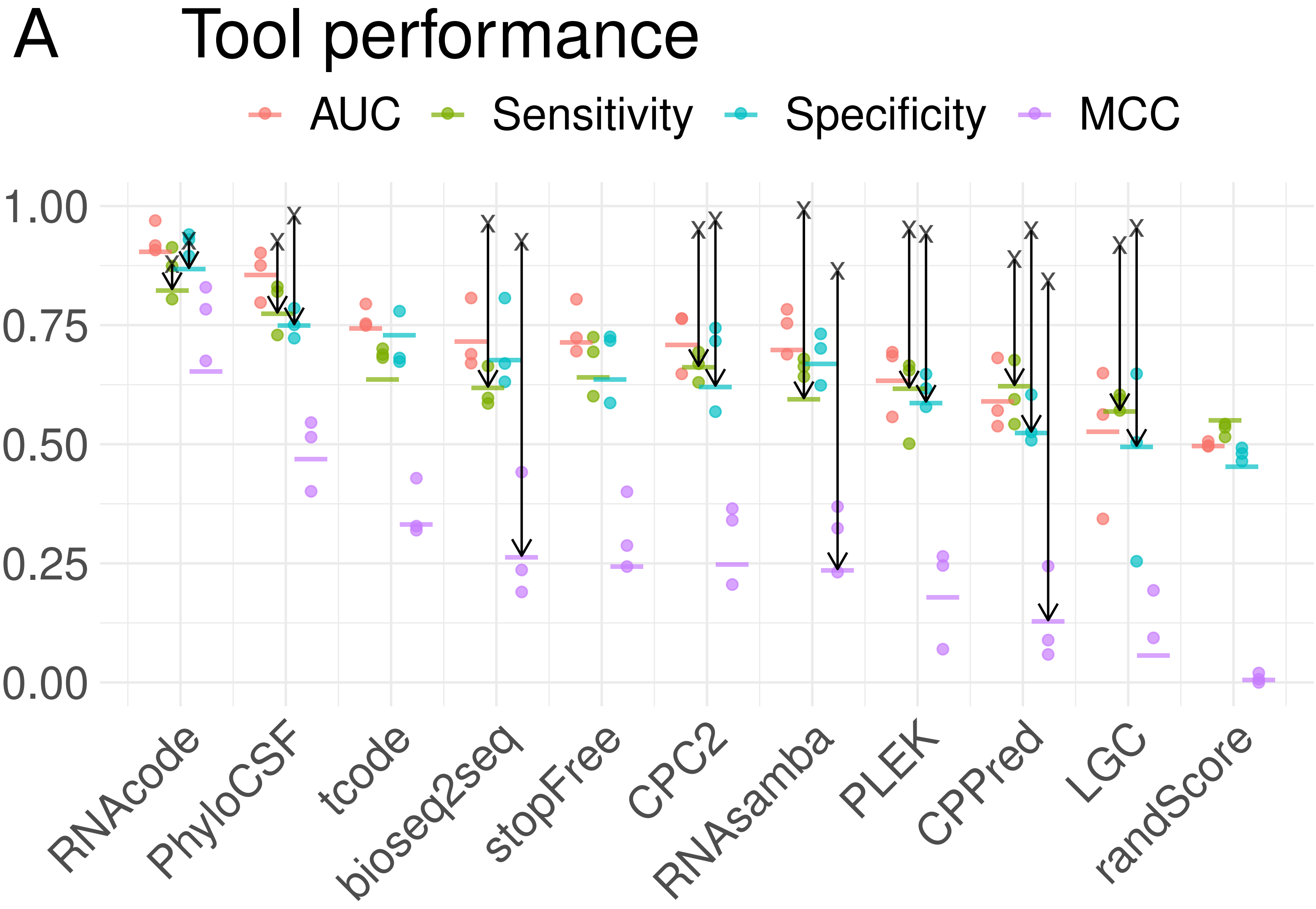
(f)

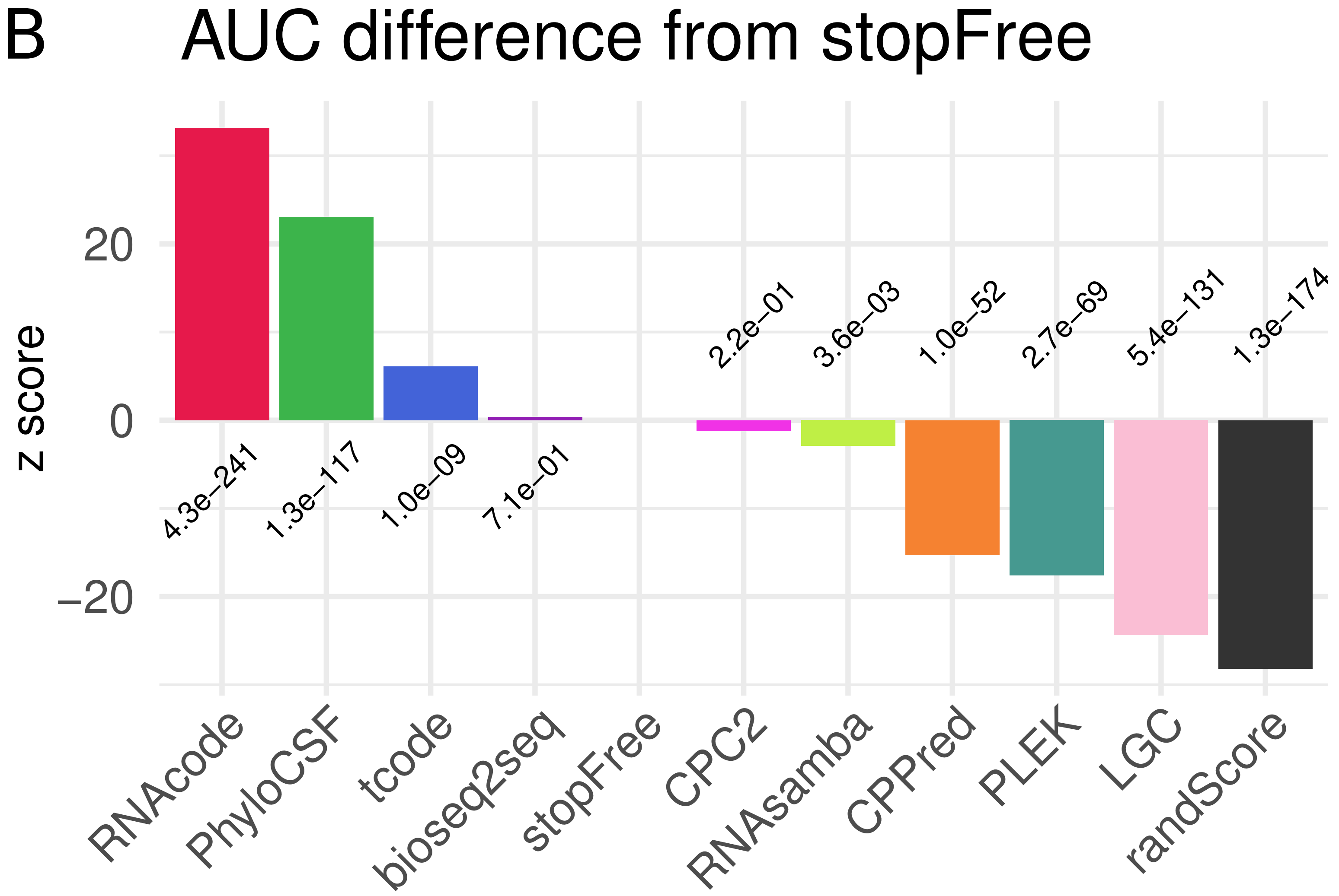
Figure S9. Score distributions for sequence-based tools RNAsamba (first row panels **a** & **b**), stopFree (second row panels **c** & **d**) and tcode (third row panels **e** & **f**). The *z* scores for coding, intergenic, and shuffled sequences have been calculated for each test clade. The distributions are depicted as histogram line graphs (left panels **a**, **c**, & **e**) and violin plots (right panels **b**, **d**, &**f**) for each clade and data type.

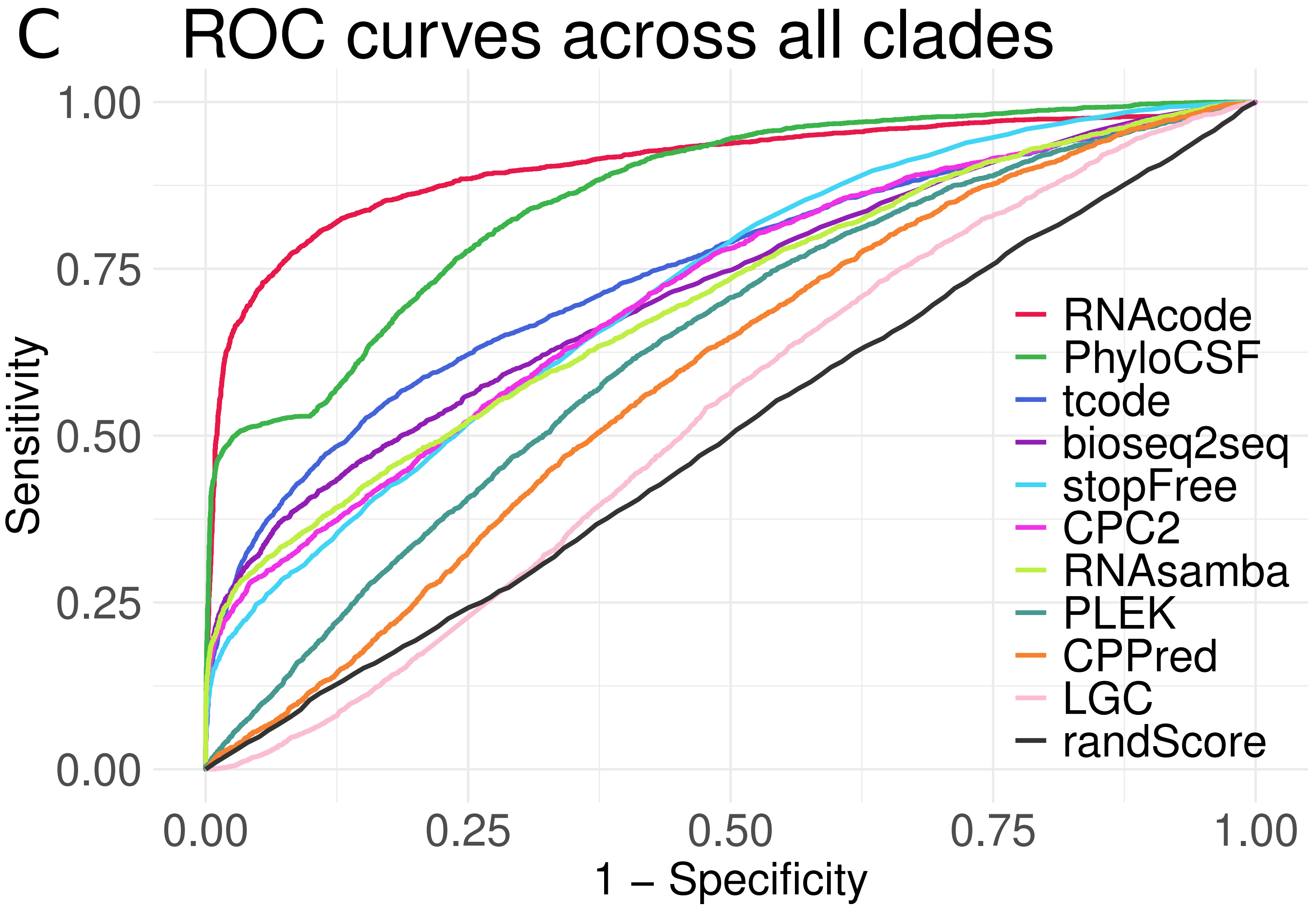

A
Intergenic, negative control
Coding exons 80-1,200 nts
Intergenic, negative control
75 nts non-coding flank
75 nts non-coding flank
B
Positive set: coding sequence
Coding seq V P P -/C/H/R H/R I/V G V F L
Felis GTCCCTCCC---CATATTGGGGTTTTCCTA
Mustela GTTCCTCCCTGTCGTATTGGGGTTTTCCTT
Equus GTTCCTCCCCATCGTGTTGGGGTTTTCCTA
Tursiops GTGCCCCCTCGCCGAATTGGGGTGTTCTTA
C
Negative set: shuffled alignment
Felis -G-TAGCTACATTGCCCCGCGATTCTTC-T
Mustela TGGTAGTTACTTTGCCCCGCGGTTCTTCTT
Equus CGATGGTTACATTGCCCCGCGGTTCTTCTT
Tursiops CGGCAGGTACATTGCCCTGCGGAGCTTTCT
D
Negative set: intergenic alignment
Felis CTTATTCTATTTATAGGTCTGATT----CT
Mustela CTTGTTCTCTGTATGTATAGGTCTGATTCT
Equus TTTATTCTCTGTATCTATGAGTCTGTTTCT
Tursiops TTTGTTCTCTGTATCTATGAGTCTGTTTCT
E
Felis catus
Prionailurus bengalensis
Felis chaus
Panthera tigris
Mustela erminea
Tursiops truncatus
Equus caballus
Epinephelus fuscoguttatus
Anopheles funestus
Kluyveromyces lactis
F
Aspergillus puulaauensis
Aspergillus sydowii
Aspergillus versicolor
Aspergillus nidulans
Aspergillus mulundensis
Aspergillus luchuensis
Aspergillus niger
Aspergillus campestris
Fusarium culmorum
Ustilago bromivora
Kluyveromyces lactis
G
Cucumis melo
Cucumis sativus
Benincasa hispida
Cucurbita pepo
Prunus persica
Cicer arietinum
Solanum pennellii
Nicotiana attenuata
Nymphaea colorata
Oryza sativa
Brachypodium distachyon

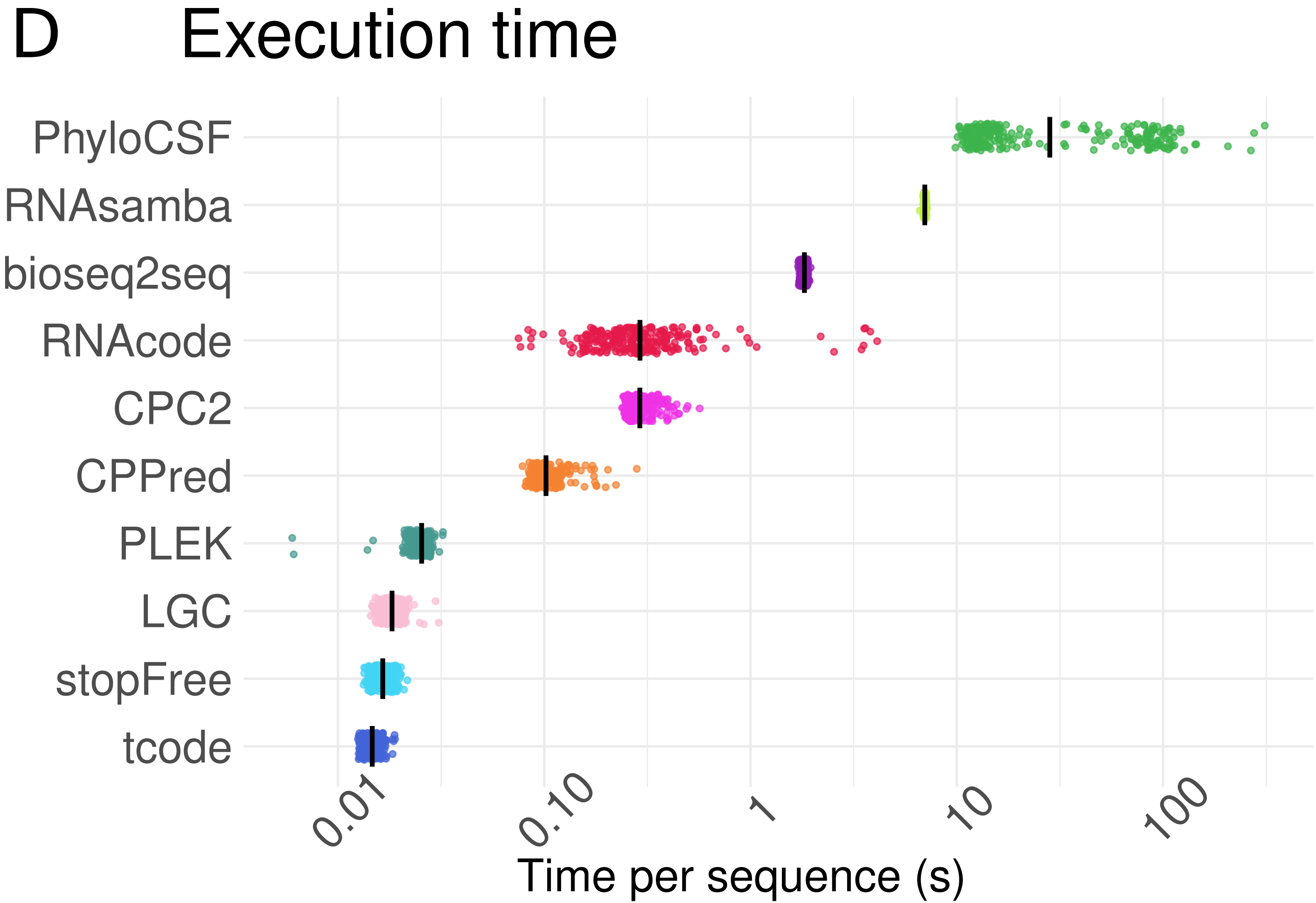

A
Number of included tools
& exclusion reasons
included
uninstallable
speciesSpecific
inaccessible
notUnique
noCodingPot.
usesHomology
infrequentlyUsed
25.0%
30.6%
25.0%
19.4%
8.3%
2.8%
2.8%
0.0%
0
1
2
3
4
5
6
7
8
9
10
11
12


B
Average citations per year
(Google Scholar, 2024)
CPC2
PhyloCSF
PLEK
tcode
RNAsamba
CPPred
RNAcode
LGC
Bioseq2seq
0
50
100
150
200


A
Tool performance
AUC
Sensitivity
Specificity
MCC
1.00
0.75
0.50
0.25
0.00
RNAcode
PhyloCSF
tcode
bioseq2seq
stopFree
CPC2
RNAsamba
PLEK
CPPred
LGC
randScore


B
AUC difference from stopFree
z score
20
0
−20
4.3e−241
1.3e−117
1.0e−09
7.1e−01
2.2e−01
3.6e−03
1.0e−52
2.7e−69
5.4e−131
1.3e−174
RNAcode
PhyloCSF
tcode
bioseq2seq
stopFree
CPC2
RNAsamba
CPPred
PLEK
LGC
randScore


C
ROC curves across all clades
Sensitivity
1.00
0.75
0.50
0.25
0.00
1 – Specificity
0.00
0.25
0.50
0.75
1.00
RNAcode
PhyloCSF
tcode
bioseq2seq
stopFree
CPC2
RNAsamba
PLEK
CPPred
LGC
randScore


D
Execution time
PhyloCSF
RNAsamba
bioseq2seq
RNAcode
CPC2
CPPred
PLEK
LGC
stopFree
tcode
0.01
0.10
1
10
100
Time per sequence (s)