

Uncertainty-aware benchmarking reveals ambiguous transcripts in mRNA–lncRNA classification

Benchmarking pipeline overview.....	2
Supplemental figure.....	2
Benchmarking dataset.....	3
Supplemental information.....	3
Supplemental methods.....	3
Coding sequence extraction.....	3
Supplemental tables.....	4
Classification tools: selection, retraining and performance.....	6
Supplemental figure.....	6
Supplemental methods.....	6
Tool's specific methods.....	6
CPAT.....	7
FEELnc.....	7
LncDC.....	8
Lncfinder.....	9
PIncPRO.....	10
mRNN.....	11
RNAsamba.....	12
LncRNA-BERT.....	13
Performance evaluation scores.....	14
Supplemental figure.....	15
Supplemental Tables.....	15
Categorization of transcripts based on confidence and agreement.....	17
Supplemental figure.....	17
Incorporation of repetitive elements information.....	18
Supplemental methods.....	18
Incorporation of non-B DNA structure information.....	20
Supplemental methods.....	20
Incorporation of RNA secondary structure features.....	21
Supplemental methods.....	21
Supplemental results.....	22
Bibliography.....	23

Benchmarking pipeline overview

Supplemental figure

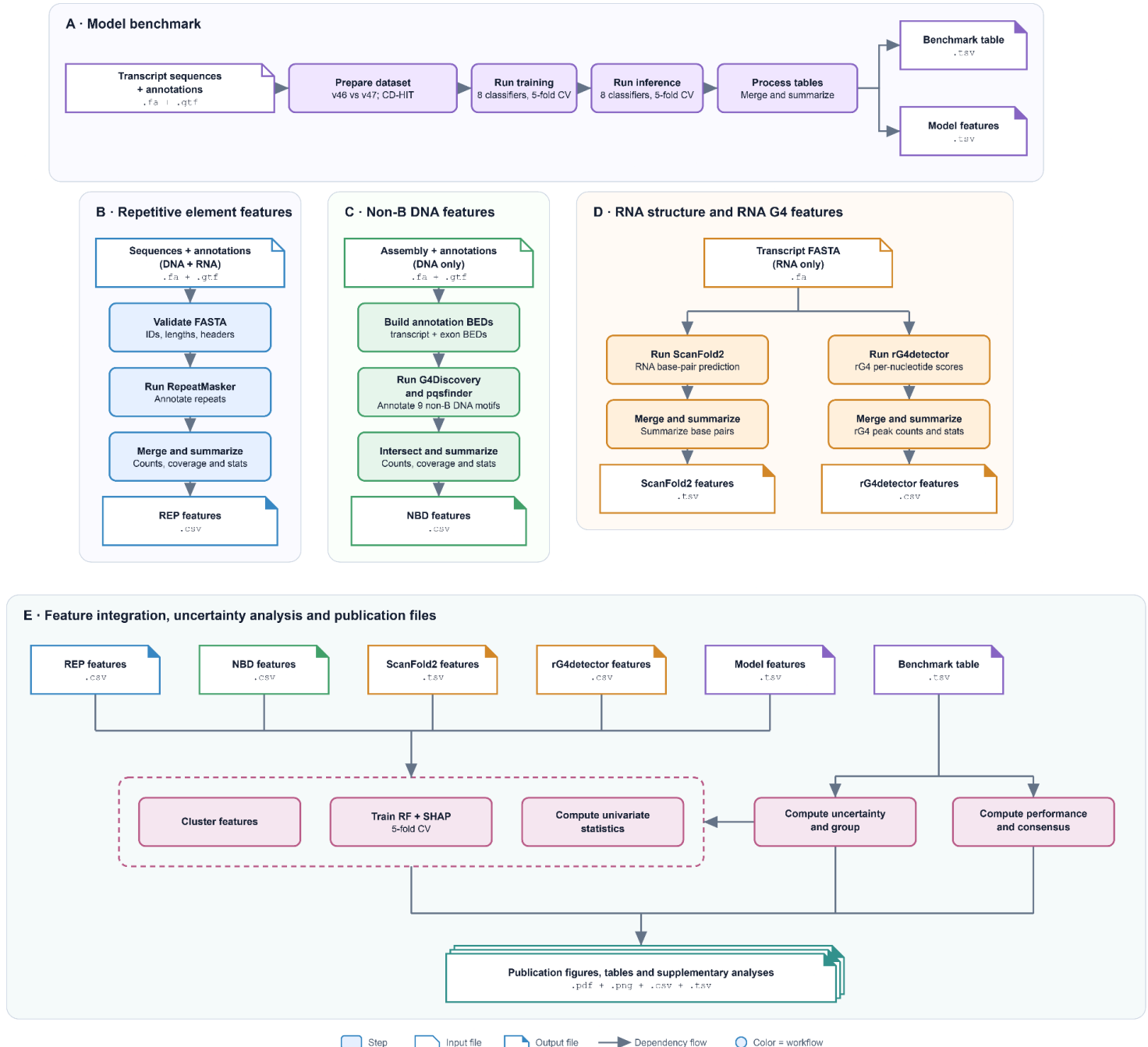


Figure S1. Benchmarking pipeline overview.

Benchmarking dataset

Supplemental information

We performed a systematic comparison between consecutive GENCODE versions (v46 and v47 for human). For each transcript, we tracked its presence and classification status across versions, categorizing transcripts into groups: (1) common transcripts with stable classification (here called “common dataset”), (2) common transcripts with changed classification, (3) newly added transcripts with assigned classification and (4) newly added transcripts without assigned classification. Additional groups were: (5) common transcripts that lost classification, (6) common transcripts that gained classification, (7) common transcripts without classification, (8) removed transcripts with classification, (9) removed transcripts without classification. Groups 2 to 9 were excluded from the current analyses.

Supplemental methods

Coding sequence extraction

The benchmarked classification tools CPAT, IncDC and LncFinder required coding sequences (CDS) of transcripts as training inputs. For protein-coding transcripts in the training set, we extracted their corresponding using the Ensembl CDS annotations.

Supplemental tables

Table S1: Transcript classification tools displayed in Figure 2A.

Tool	Year	Reference
CPC	2007	Kong et al., 2007 [1]
PORTRAIT	2009	Arrial et al., 2009 [2]
PhyloCSF	2011	Lin et al., 2011 [3]
CPAT	2013	Wang et al., 2013 [4]
CNCI	2013	Sun et al., 2013 [5]
IncRScan	2015	Sun et al., 2015 [6]
COME	2017	Hu et al., 2017 [7]
FEELnc	2017	Wucher et al., 2017 [8]
CPC2	2017	Kang et al., 2017 [9]
PLncPRO	2017	Singh et al., 2017 [10]
longdist-SVM	2017	Schneider et al 2017 [11]
mRNN	2018	Hill et al., 2018 [12]
IncRNAnet	2018	Baek et al., 2018 [13]
LncADeep	2018	Yang et al., 2018 [14]
CPPred	2019	Tong et al., 2019 [15]
PredLnc	2019	Liu et al., 2019 [16]
LncFinder	2019	Han et al., 2019 [17]
RNAseba	2020	Camargo et al., 2020[18]
LncDC	2022	Li et al., 2022 [19]
LncRNA-BERT	2025	Romejin et al., 2025 [20]

Table S2: Number of transcripts by class and version, as displayed in Figure 2B.

From (v46)	To (v47)	Count
new transcripts	lncRNA	131,424
protein-coding	protein-coding	111,838
no class	no class	82,231
lncRNA	lncRNA	59,674
new transcripts	protein-coding	331
lncRNA	removed transcripts	187
new transcripts	no class	52
lncRNA	no class	43
no class	protein-coding	26
lncRNA	protein-coding	23
protein-coding	removed transcripts	20
no class	removed transcripts	11
protein-coding	no class	9
no class	lncRNA	7
protein-coding	lncRNA	1
	Total	385,877

Classification tools: selection, retraining and performance

Supplemental figure

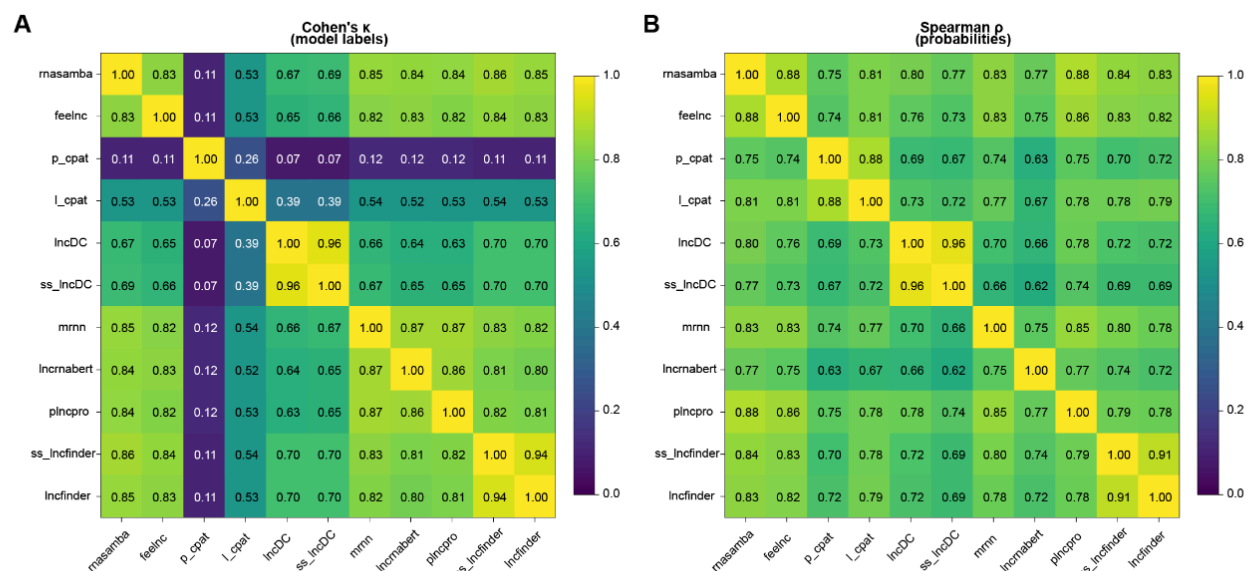


Figure S2. Diversity of model predictions. (A) Heatmap representing Cohen's kappa statistic on the similarity of predicted labels. (B) Heatmap representing spearman correlation coefficient of model coding scores.

Supplemental methods

Tool's specific methods

First, tools were required to have publicly available source code, with a preference for widely used and actively maintained methods, ensuring the reproducibility and long-term accessibility of the benchmark. Second, inclusion required support retraining directly or with minimal modifications, allowing all models to be evaluated with any custom benchmarking dataset. Third, since uncertainty analyses are grounded on approximating model scores to coding probabilities, benchmarked tools must report classification as a continuous score between 0 and 1. Fourth, selected models span a diversity of algorithm families (including logistic regression, support vector machines, tree algorithms, neural networks and transformer-based architectures) so that model disagreement reflects real uncertainty rather than simple correlated errors arising from methodological similarities. Finally, a substantial number of these architectures were

required to be feature-based, providing the basis for post-hoc interpretability analyses of the discriminative features in different uncertainty regimes.

CPAT

The Coding Potential Assessment Tool (CPAT) is a logistic regression model that uses four sequence-based features to classify transcripts [4]. These features are (1) maximum ORF length, (2) ORF coverage (or ORF-to-transcript length ratio), (3) Fickett Score (a metric summarizing codon usage distribution) (Fickett, 1982), and (4) hexamer score (usage bias of adjacent codons). The original model was trained with 10 000 protein-coding transcripts from the RefSeq database [21] and 10 000 lncRNAs from Gencode [22]. The CPAT model was evaluated by 10-fold cross-validation, and a probability cutoff of 0.364 was selected to maximize both sensitivity and specificity when labeling transcripts. Finally, classification performance was tested on an independent set containing 4000 protein-coding transcripts from RefSeq and 4000 lncRNAs from a different curated dataset (Cabili et al 2011), achieving a sensitivity of 0.96 and a specificity of 0.97.

We used CPAT version 3.0.5 to train a custom model following the instructions in the original publication and the online documentation. Since version 3.0.0, CPAT allows users to select the ORF with highest coding probability instead of the longest ORF for ORF feature calculation. We therefore trained models with both options and selected the best-performing one (Table S8). A reference hexamer frequency table was built using only the training dataset as inputs. We ran a 10-fold cross validation and found an optimal probability cutoff value of 0.410, slightly higher than the original 0.364. The final logistic regression model was trained on the full training dataset and model performance was evaluated on the common test set after assigning classification labels based on the cutoff (i.e., $p < 0.410 \rightarrow$ lncRNA; $p \geq 0.410 \rightarrow$ protein-coding).

FEELnc

The Flexible Extraction of lncRNAs (FEELnc) tool is a pipeline for lncRNA detection and annotation [8]. It includes the FEELnc codpot module (hereon called simply FEELnc) which is a Random Forests (RF) model that calculates a coding potential score based on three sequence features: (1) transcript length, (2) ORF coverage and (3) Kmer Scores (summarizing multi kmer frequency differences between coding and non-coding transcripts). The transcript set used for

training FEELnc was extracted from Gencode v24 and was composed of 5000 coding transcripts (those with biotype 'protein_coding') and 5000 lncRNA transcripts (with biotypes 'lincRNA' and 'antisense'). A held-out test set was also created using 5000 coding and 5000 lncRNA transcripts. Importantly, only one transcript per gene was used, thereby avoiding sequence redundancies within and between training and testing datasets. The method for extracting ORFs and the number of k-mer scores calculated were evaluated to find the best-performing combination. The optimal parameters were (1) to use the longest ORFs with either a start codon, a stop codon or both (ORFs of type 3, following the notation of the FEELnc publication); and (2) to calculate multi k-mer scores of k-mers of k in {1, 2, 3, 6, 9, 12}, instead of individual ones. Finally, and like CPAT, a 10-fold cross-validation was used to calculate the optimal coding probability cutoff, which is automatically used by the tool to label transcripts. FEELnc was evaluated on the test set together with other classification tools, producing a sensitivity of 0.923, specificity of 0.915 and an F1-score of 0.919. It is important to note that, instead of providing a trained model, the FEELnc tool trains a RF classifier on the go using the provided training set.

We used FEELnc version 0.2.1 to train and evaluate a RF model on our dataset. We ran FEELnc following the optimal parameters provided in the publication, including the use of 'type 3' ORFs and the calculation of multi k-mer scores. The probability cutoff found was 0.4202.

LncDC

LncDC is a gradient boosting classifier based on the XGBoost library [19]. This model leverages a set of sequence-intrinsic, secondary structure and protein features. Interestingly, LncDC predicts ORFs in 4 distinct ways (called ORF types) and calculates sequence and protein features for all of them. The LncDC training and testing datasets were constructed as follows: Protein coding transcripts were collected from RefSeq release 99 and lncRNAs from Gencode v34. Transcripts were filtered out by three criteria: (1) length < 200 nt, due to the definition of lncRNAs as transcripts longer than 200 nt (2) length > 20000 nt, due to the exponential computational cost of SS calculation and (3) presence of nucleotides other than 'ACGT'. Then, a held-out test set was constructed with 10000 protein-coding transcripts and 10000 lncRNAs. The rest of the sequences were scanned with CD-HIT to remove those with an identity higher than 90%, limiting sequence redundancy. The remaining 37,698 mRNAs and 34,539 lncRNAs were used to build the training dataset. In addition, to balance the number of transcripts in both classes, a set of synthetic lncRNA sequences was created using the SMOTE technique, which

recursively creates sequences by selecting random pairs of instances in the feature space and choosing a random point between them.

A set of 57 features was evaluated by training an XGBoost model and performing random feature elimination with cross-validation (RFECV). A subset of 28 features was found to have the maximum accuracy (0.9823) while minimizing the number of features, avoiding potential feature redundancy and overfitting. This final set contained (1) sequence features, such as the GC-content, the Fickett score, and metrics of different ORF types (maximum ORF length, ORF coverage, hexamer scores and relative codon bias); (2) secondary structure features, including the GC-content of paired nucleotides and the sequence and secondary-structure (SASS) k-mer scores for k values in [1,5], and protein features, like the isoelectric point, molecular weight, aromaticity and instability of the translations of different ORF types.

The final model was trained and evaluated with the data sets described above. LncDC achieved a sensitivity score of 0.9812, specificity of 0.9786 and an F1-score of 0.9799.

We trained a LncDC model using version 1.3.6 with our custom dataset. Since the tool allows for training either with or without SS features, we built those models and evaluated their performance on our held-out data set. Like in the LncDC publication, the model integrating SS features provides a marginal performance improvement, so we continued working with this version (Table S8). Importantly, model training produces custom files that are key for later model inference, including an hexamer table, an imputer for over-sampling of the minority class and a scaler to standardize feature values as z-scores using training data feature distribution.

Note that we included minor modifications in the source code of LncDC to allow it to load precalculated secondary structure sequences, reducing training and inference time. These modifications can be found in our fork of the LncDC code (<https://github.com/cbib/LncDC>, commit ID 8cbce76)

Lncfinder

Lncfinder is a toolbox to evaluate transcript features and train transcript classifiers [17]. It distributes a trained support vector machine (SVM) classifier, hereby called Lncfinder. Lncfinder incorporates a total of 19 features, including (1) sequence-intrinsic features (the logarithmic

distances (LDs) of hexamer composition, the length and the coverage of ORFs), (2) secondary structure features (MFE, frequency on unpaired nucleotides followed by a paired nucleotide or UP frequency, and the LDs of different encodings of secondary structure); and (3) physicochemical features based on the fast fourier transform (FFT) of electron-ion interaction pseudo-potential (EIIP) (the signal at $\frac{1}{2}$ position, the signal to noise ratio and the signal quantile statistics min, Q1, Q2 and max). Key features within each group were selected after RFECV or a custom selection algorithm to later validate the global performance of combining all 19 features.

The LncFinder publication uses datasets described elsewhere (Achawanantakun et al., 2015). Briefly, Human dataset A (or H1 in the original publication) contains 15308 protein-coding transcripts and 4586 lncRNAs in the training set, all extracted randomly from single genes from the Gencode database. The testing set is made of 4000 transcripts of each class, with none of them sharing genes with training transcripts. Additionally, Human dataset B (or H2) is a modified version of the CPAT dataset (see above), with a training set reduced to 9929 mRNAs and 9066 lncRNAs due to database updates, and an equivalent testing set. Feature selection and architecture comparison was performed by 10-fold CV using the training set of Human dataset A, and final model training and benchmarking used the training and testing sets of Human dataset B, respectively. Evaluation of LncFinder SVM resulted in a sensitivity of 0.9544, specificity of 0.9150 and F1-score of 0.9360.

We trained a custom SVM using LncFinder version 1.1.6. Due to the computational constraints of the RNAfold tool used for SS calculation, sequences longer than 30000 nucleotides were excluded for the analysis (Table S19). Given that the SS predicted by RNAfold may not be the actual molecular structure and to include the long transcripts, we evaluated the performance of the a LncFinder model without SS information. The metrics were slightly reduced, so we kept the SS model for further experiments (Table S8).

PlncPRO

PlncPRO is a transcript classification tool, originally developed for plants, based on random forests[10]. PlncPRO is trained on a set of 71 features, including:

- 64 trimer frequencies

- 4 metrics extracted from BLASTX hits against SWISS-PROT. These were (1) number of hits (N); (2) significance score (S) calculated as the sum of negative natural logarithm of the e-value for each BLASTX hit; (3) the sum of bitscores of all hits (B); and (4) frame entropy (F), calculated as the Shannon entropy of the probability that a hit falls within a given reading frame.
- 2 ORF-based features from Framefinder, including the ORF coverage and the Framefinder score (FF-score).

The PlncPRO program trains 5 independent RF classifiers and keeps the one with highest out-of-bag score, a metric that estimates generalization error. Finally, the selected model is used for transcript classification. The PlncPRO publication describes the training and evaluation of multiple models trained on transcripts of a range of plant species. Moreover, they report the performance of a similar model trained and tested on Human transcripts. This model achieves accuracies of 0.9434 and 0.9372 on two Human test sets.

Model retraining was performed as per the PlncPRO instructions. However, due to an unidentified bug, BLASTX features were not properly computed for the training set, which may negatively impact classification performance.

mRNN

mRNA RNN (mRNN) is a modified recurrent neural network (RNN) model that uses gated recurrent units (GRU) to learn sequential information from transcripts and classify them [12]. mRNN encodes sequences as embedding vectors and uses dropout regularization to reduce overfitting. This model was trained and evaluated on a series of datasets created from Gencode v25 transcripts: (1) a test set with 500 mRNAs and 500 lncRNAs longer than 200nt extracted from individual genes, (2) a “challenge” set consisting of mRNAs with ORFs shorter than 150 nt and lncRNAs longer than 200 nt containing a non-coding ORFs of at least 150 nt, (3) a “long 5’ UTR set” comprising the 500 mRNAs with longest 5’UTRs; from the remaining sequences, (4) a validation set with 500 coding and 500 non-coding transcripts, (5) a training set only including sequences between 200 and 1000 nt, with a total of 16000 transcripts of each class; and (6) an augmented pre-training dataset constructed by introducing 10 mutated copies of training transcripts, each with one random 1-nt insertion. Pre-training was performed on sets 6 and 4, model training was done with datasets 5 and 4, and sets 1, 2 and 3 were used in final model evaluation. It is important to note that pre-training set augmentation by random insertions and

restriction of training sequence lengths were reported to provide improvements in classification performance. The training process entailed (1) pre-training 30 models with distinct augmented datasets for 4 epochs, (2) evaluating pretrained models on the validation dataset to select the top 6, (3) further train the best models on the non-augmented dataset for 10 epochs during 10 independent sessions and (4) select the top 5 final models, keeping only one resulting model per pre-trained dataset. Finally, transcript classification is performed either by using the best single model (mRNN) or integrating the top 5 models by averaging model weights (mRNN ensemble). The ensemble model achieved higher scores in all metrics analysed, including specificity, sensitivity and F1-score higher than 0.95 (average of 100.000 bootstrap trials with replacement).

We modified the published version of the mRNN code to increase the maximum allowed transcript length, which was hardcoded and caused an infinite loop if longer sequences were used. Code modifications can be found in our fork (<https://github.com/cbib/mRNN>). We used this modified version of mRNN (commit 41fc34a) to train an ensemble classifier. The common training dataset was further split into pre-training and validation datasets, with the latter containing the same number of transcripts as the testing set. In order to keep an equivalent training set for all tools evaluated in this study, the training set was not filtered by transcript size. Then, data augmentation, model pre-training, pre-trained model selection, model training and final model evaluation were performed following the same protocol as the original publication.

RNAseamba

RNAseamba is a deep learning model that incorporates the IGLOO architecture [18]. Its main innovation is the combination of two independent branches: B1 branch, that uses the nucleotide sequence (trimmed to 3,000 nucleotides) to capture coding signals independent of ORF identification; and the B2 branch, that integrates the amino acid sequence encoded by the longest ORF (truncated at 1,000 aa) together with sequence features, including nucleotide k-mer frequencies, amino acid relative frequencies and ORF length. These two branches are combined through a learned attention parameter that dynamically weights their relative contribution, which is particularly important for transcripts where no ORF was identified.

The RNAseamba authors published two trained models: one trained on full transcript sequences, and a second one trained on both full-length and truncated transcripts. Train/test sets were

extracted from previous publications, including CPC2, FEELnc, mRNN and CPPred. Sequence redundancy was limited by running MMseq with 90% homology and 90% coverage filters. Model evaluation was performed against CPC2, CPAT, FEELnc, lncRNA-net and mRNN on several datasets, and RNAsamba scored the highest area under the precision-recall curve (AUC) for all of them, with AUC values between 94.16% and 99.67%.

We trained an RNAsamba model using full transcripts following the original instructions. The choice to use full sequences instead of truncated transcripts was motivated by the fact that our benchmark is based on reference datasets instead of on experimental transcript models that would increase fragmentation.

lncRNA-BERT

lncRNA-BERT is a BERT-based RNA language built on the transformer encoder architecture with a binary classification head [23]. It supports two sequence encoding strategies:

- K-mer tokenization (k=3): overlapping trinucleotide tokens from the raw sequence (64-token vocabulary).
- Convolutional Sequence Encoding (CSE) (k=9): a learnable convolutional layer that embeds raw nucleotide sequences into a continuous token space, providing stronger compression that allows longer transcripts to be classified.

Since transformer architecture is limited by the context size, lncRNA-BERT trims the sequence end to make it fit.

The model training is composed of two steps:

- Pre-training (masked language modeling): The model learns general RNA sequence representations from GENCODE v46, RefSeq v255, and NONCODE v6 (a total of 297,724 coding and 238,470 non-coding sequences). 5% of GENCODE is held out for validation.
- Fine-tuning (binary classification task): The pre-trained model is fine-tuned for coding/non-coding classification on 101,270 coding and 48,785 lncRNA sequences from GENCODE and RefSeq (NONCODE excluded to maximise data reliability). Importantly, sequence redundancy in the fine-tuning set was limited by using CD-HIT with a 90% identity threshold.

lncRNA-BERT models were evaluated on three testing datasets: (1) GENCODE/RefSeq set, (2) CPAT test set and (3) the multi-species RNACHallenge dataset of transcripts difficult to classify.

Both k-mer tokenization and CSE encoding versions were evaluated, with the following respective F1 scores: (1) 0.940 and 0.943, (2) 0.963 and 0.947; and (3) 0.235 and 0.242. Importantly, while the performance for the RNACChallenge dataset is significantly lower to the other datasets, lncRNA-BERT models achieved the best performance over all models compared in their study (i.e., CPAT, LncFinder, PredLnc, LncADeep, RNAsamba and mRNN).

For our benchmark, we used the pre-trained model with the k-mer tokenization (k=3) and fine-tuned each validation fold with its corresponding training dataset. Since the fine-tuning step requires a validation dataset, similarly to mRNN, we held-out a subset of transcripts from the training set for this purpose.

Performance evaluation scores

To benchmark the eight different ML models, several scores were used based on the four confusion matrix values: true positive (TP), true negative (TN), false positive (FP) and false negative (FN):

$$Precision = TP / (TP + FP)$$

$$Recall = TP / (TP + FN)$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

$$Balanced\ accuracy = (True\ positive\ rate + True\ negative\ rate) / 2$$

Where the true positive rate is the same as recall (or sensitivity), and true negative rate (TNR or specificity) is:

$$TNR = TN / (TN + FP)$$

Supplemental figure

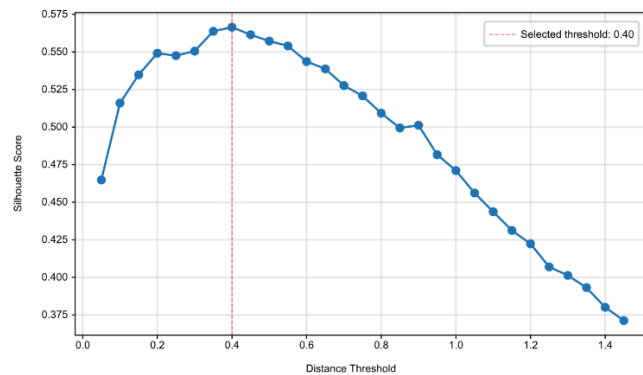


Figure S3. Silhouette score of feature clusters. (A) Silhouette score was calculated for incremental distance thresholds, and the highest was retained as optimal to define the number of clusters (220 clusters at a distance of 0.4)

Supplemental Tables

Table S17: Number of features after each processing step							
Step	ML Set	Rep Set (RNA)	Rep Set (DNA)	NBD Set	rG4 Set	ScanFold Set	Total
Numerical features	128	170	170	199	4	63	734
Non-constant	128	157	157	199	4	63	679
Continuous	128	142	142	189	3	63	667
Representative	53	36	35	52	1	42	220*
Categorical	0	15	15	10	1	0	41
Final set	53	51	50	62	2	42	261*
<p>TSV files from each pipeline were parsed to extract numerical features. Features with constant values across all transcripts were removed. Features were separated into continuous features (which were reduced to one representative feature per cluster after hierarchical clustering) and categorical features. The final feature set contains all representative continuous features plus the categorical features. One of the resulting feature clusters (C20) is composed of features from both the Rep DNA set and NBD set. Since the representative features can be chosen arbitrarily, here it is only counted in the calculated totals (*) instead of a specific feature set.</p>							

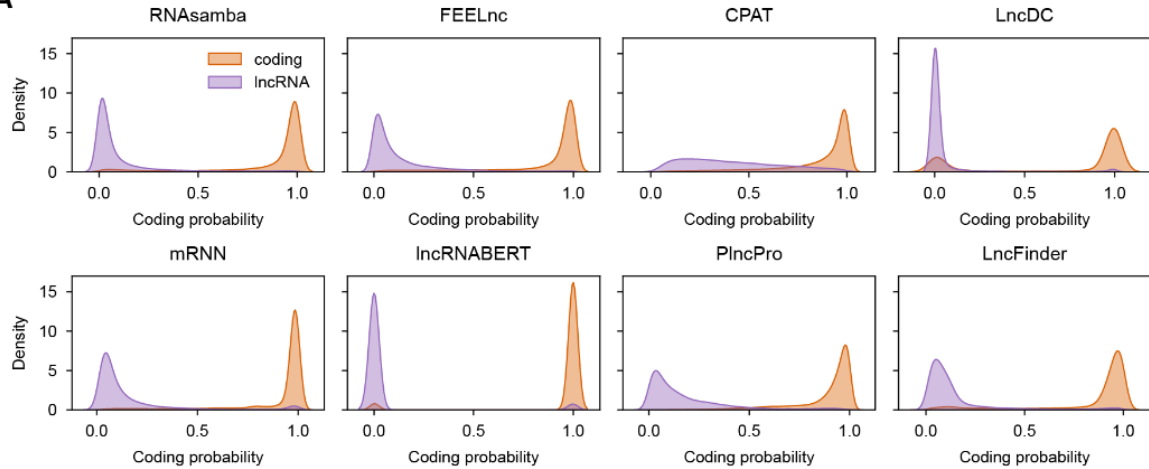
Table S19: Transcripts in the “common CDHIT” set without missing classification labels		
tool	count	reason
total	112,264	(All transcripts in the common cdhit dataset)
total no class	612	(All transcripts that were not classified by all tools)
ss_IncDC	529	Sequence shorter than 200 nt, containing non-ACGT characters or longer than 20,000 nt.
l_cpat	180	ORF not found or sequence shorter than 75 nt.
feelnc	5	ORF not found
ss_Incfinder	15	Sequence longer than 30,000 nt.

Tables S3, S4 and S6 to S16 are available in the Supplemental Tables file.

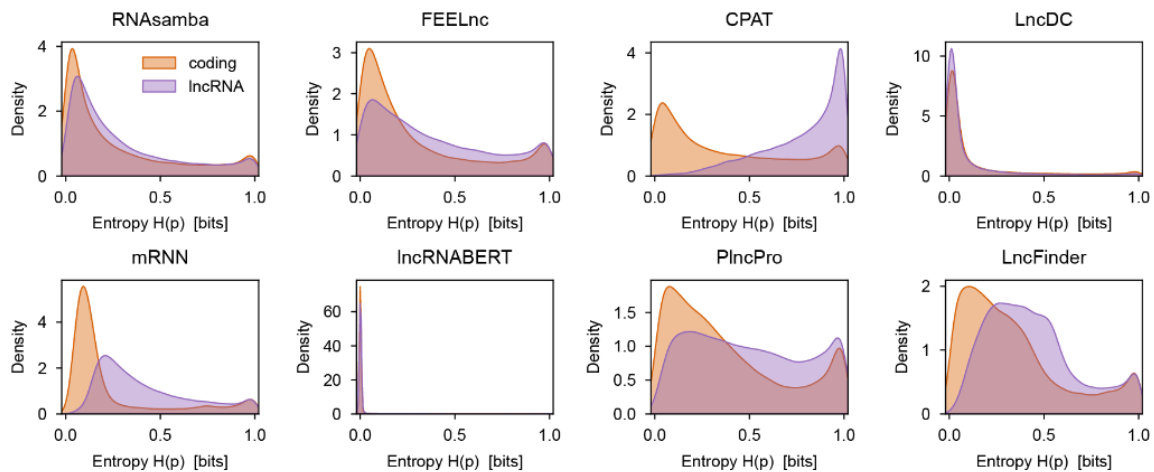
Categorization of transcripts based on confidence and agreement

Supplemental figure

A



B



C

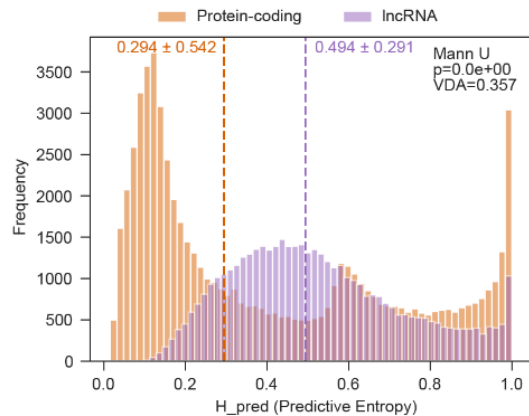


Figure S4. Model probabilities and entropy distributions. (A) Distribution of model probabilities, where 1 = protein-coding and 0 = lncRNA. (B) Distribution of model entropy, where 0 = full certainty and 1 = full uncertainty. (C) Distribution of predictive entropy (H_{pred}).

Incorporation of repetitive elements information

Supplemental methods

Gene body sequences (or unspliced transcript sequences) were extracted using annotations and genome fasta files from GENCODE v47. First, transcript coordinates were parsed from the GTF annotation, retraining transcript-level records only on a 0-based BED coordinates file. These coordinates correspond to the full genetic span of the unspliced transcript. The sequence was extracted from the reference fasta file using the strand-aware command *bedtools getfasta -s*. Spliced transcript sequences were used directly from the GENCODE transcript fasta files.

RepeatMasker v4.2.1, with its default human database (Dfam version 3.9) as reference, was used to annotate the repetitive elements present in both transcripts (RNA) and gene bodies (DNA), separately. RepeatMasker was run with the following parameters: species set to “human”, sensitive mode (-s) and parallel processing. The Gencode FASTA headers were trimmed to the first pipe character (“|”) to ensure compatibility.

RepeatMasker output files (“.out”) containing hits were parsed. To keep a single hit annotation per coordinate, overlapping hits were resolved following a hierarchical scoring system. RM reports some elements that overlap with a higher-scoring hit with an asterisk in the last column of the “.out” file. For overlapping pairs, we removed (1) hits tagged with an asterisk by RM; if neither had an asterisk (2) hits with lower Smith-Waterman alignment score (SW score); if scores were equal, (3) hits with lower percentage overlap. If overlaps remained, we resolved them by keeping the first reported element.

Then, using a derivative of the Dfam classification, hits were separated into four groups detailed in the following table:

Table S20: Groups of repetitive elements used in the feature extraction pipeline	
Group	Class

Table S20: Groups of repetitive elements used in the feature extraction pipeline	
Transposable Elements	DNA, LINE, LTR, PLE, RC, Retroposon, SINE and srpRNA*
Interspersed repeats and low complexity regions	Satellite, Simple_repeat and Low_complexity
Pseudogenes	rRNA, scRNA, snRNA and tRNA
Unknown	Unknown
<p>* Due to the absence of clear classification of srpRNAs in the classification tree of Dfam v3.9 (https://www.dfam.org/releases/Dfam_3.9/infrastructure/TEClasses.tsv), we included srpRNAs as a standalone class within the Transposable Elements class. This choice was informed by the fact that all srpRNA elements are 7SLRNA in Dfam family DF000000016, which is included in the Alu class.</p>	

Then, a comprehensive set of features was extracted using a custom Python pipeline. Feature extraction proceeded hierarchically at three levels:

Hit-level features: individual RepeatMasker hits were characterized by SW score, divergence from the consensus sequence, insertions and deletions (all RM features). We further calculated hit length and reference coverage (proportion of the consensus repeat covered by the hit).

Element-level features: RepeatMasker reports fragmented hits that represent nested insertions or incomplete elements. Fragments belonging to the same original element were aggregated using RepeatMasker's hit identifiers. For each reconstructed element, we computed total length, total reference coverage, number of fragments and fragmentation status (a binary indicator of having >1 fragment). Element-level classifications were performed for seven major TE classes: LINE (Long Interspersed Nuclear Elements), SINE (Short Interspersed Nuclear Elements), LTR (Long Terminal Repeats), DNA transposons, RC (rolling-circle transposons), Retroposons and PLE (Penelope-like elements). Element classifications included family resolution, capturing Alu, MIR, L1, L2 and ERV subfamilies among others. Elements were additionally classified as “young” (L1HS, L1PA1–2, AluY, SVA_F, HERVK) or “ancient” (divergence >20%).

Transcript-level features: All elements overlapping each transcript were aggregated to generate transcript-level summary statistics. For each transcript, we computed: (1) count-based

features including total number of hits, number of fragmented elements, fragmentation ratio and diversity metrics (number of unique subfamilies, families and classes); (2) length-based features including total TE coverage (bp and percentage), and minimum/mean/maximum (min/mean/max) values for hit lengths and reference coverage values; (3) divergence-based features including min/mean/max values for SW scores, sequence divergences, deletions and insertions; (4) class-presence features including hit counts and presence/absence binary indicators for each major class and family; and (5) spatial features including gap statistics (min/mean/max and standard deviation of inter-element distances) and hit density (hits per kilobase).

Additionally, we calculated features relative to transcript length (hit length, min/mean/max hit length, and gap stats) and per kilobase (total hit count, class/family hit count and unique class/family count). All 158 extracted features are comprehensively detailed in Table S3.

Incorporation of non-B DNA structure information

Supplemental methods

Non-B DNA structural motifs were identified on the GRCh38 genome reference assembly using two tools. The *non-B_gfa* (gfa) pipeline (https://github.com/abcsFrederick/non-B_gfa), which generates annotation files for A-phased repeats (APR), direct repeats (DR), inverted repeats (IR), mirror repeats (MR), short tandem repeats (STR), and Z-DNA. Triplex-forming motifs (TRI) were defined as the subset of MR predictions flagged as triplex-prone by the gfa output. G-quadruplexes (G4) were predicted on GRCh38 using G4Discovery [24] with default parameters, reporting both pqsfinder [25] and G4Hunter [26] scores.

All motif calls were intersected with GENCODE v47 transcript intervals (genomic coordinates) using bedtools intersect (v7.3.0), retaining all pairwise overlaps (-wa -wb). The resulting files were fed into a feature extraction pipeline that (1) clips NBD motifs extending beyond transcript genomic boundaries, (2) resolves overlapping elements of the same motif class, (3) computes a panel of features for each motif class and (4) computes cross-motif gap statistics. Features per motif included:

- Presence indicator
- Hit count

- Min, max, mean and standard deviation of motif length.
- Total motif length, both before and after overlap resolution.
- Min, max, median, mean and standard deviation of inter-motif gaps
- Relative features as percentage of the transcript length and counts per kilobase

The full set of 178 extracted features is comprehensively detailed in Table S4.

Incorporation of RNA secondary structure features

Supplemental methods

Putative RNA G-quadruplex (rG4) forming regions were predicted with rG4detector (<https://github.com/OrensteinLab/rG4detector>), a convolutional neural network (CNN) model trained on rG4-seq data. For each transcript, *predict_fasta.py* was run in detection mode (-d) to produce a per-nucleotide sliding-window rG4 propensity score. Per-transcript rG4 features were derived by calling peaks on these scores using 1.56 as the optimal score cutoff calculated in the rg4detector article ([27], see their Supplementary figure S4) Each transcript's peaks were reduced to four features: number of peaks, peak density (peaks per nucleotide), the mean relative position of peaks and a binary peak-presence indicator.

Local secondary-structure was assessed using the ScanFold2.0 pipeline (<https://github.com/moss-lab/ScanFold2.0>), ran with default parameters for window size, sliding step, folding temperature and nucleotide shuffling: `-w 120 -s 1 -t 37 --suffle mono`. This pipeline is composed of two main steps: first, ScanFold-Scan scans each transcript with a sliding window, folds each window with ViennaRNA, and compares its minimum free energy (MFE) against a null distribution from shuffled randomized windows to derive a per-window Z-score and ensemble diversity (ED); second, ScanFold-Fold combines results into a consensus base-pairing structure with associated per-nucleotide information: the pairing partner (or unpaired flag), whether that partner may pair elsewhere; and the average MFE, Z-score and ED from all the windows supporting the structure of that nucleotide. We computed the following per-transcript features:

- Summary statistics (min, mean, median, max) of average Z-score, MFE, and ED
- Fractions of nucleotides that: are paired, compete for alternative structures, have an average Z-score below -1 or -2.
- Relative position of the global Z-score, MFE and ED minimum value.

- After dividing each transcript into 10 equal-length fragments, the fraction of the total nucleotides that are paired, competing, and low-Z-score in each fragment.

The full set of 4 RNA G4 features and 63 RNA secondary structure features are detailed in Table S7.

Supplemental results

To identify sequences prone to RNA G-quadruplex formation in our dataset, we compared the predictions from G4Discovery, a tool combining pqsfinder and G4Hunter that we already used for DNA G4 prediction; and rg4detector, a deep learning-based method trained on real rG4-seq data. Applied to our transcript set, these tools showed overlap on G4 presence prediction per transcript. However, when looking at the positional distribution of predicted rG4s, the tool differed: G4Discovery-predicted motifs showed a distribution skewed towards the end of the transcript, with slight differences for protein-coding and lncRNA transcripts. On the other hand, rg4detector predictions were enriched both at transcript start and ends, with lower rG4 density in the middle. Given this different distribution, we decided to use rg4detector for predicting RNA G4s as it is trained on experimentally-derived data.

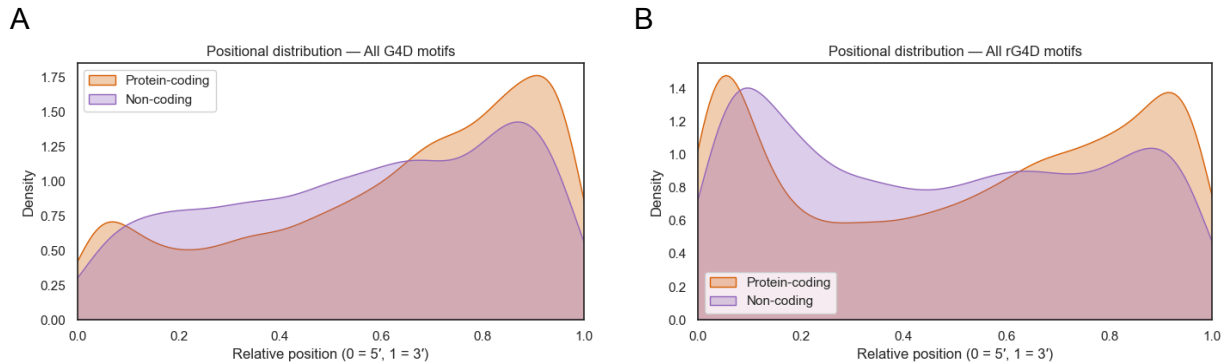


Figure S5: Positional distribution of RNA G-quadruplexes in protein-coding and lncRNA transcripts predicted by G4Discovery (A) and rg4detector (B).

Interestingly, both tools predicted lower mid-transcript rG4 density in protein-coding transcripts compared to lncRNAs, consistent with the hypothesis that coding regions select against G4 structures.

Bibliography

1. Kong L, Zhang Y, Ye Z-Q, Liu X-Q, Zhao S-Q, Wei L, et al. CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. *Nucleic Acids Res.* 2007;35:W345. <https://doi.org/10.1093/nar/gkm391>
2. Arrial RT, Togawa RC, Brigido M de M. Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus *Paracoccidioides brasiliensis*. *BMC Bioinformatics.* 2009;10:239. <https://doi.org/10.1186/1471-2105-10-239>
3. Lin MF, Jungreis I, Kellis M. PhyloCSF: a comparative genomics method to distinguish protein coding and non-coding regions. *Bioinformatics.* 2011;27:i275–82. <https://doi.org/10.1093/bioinformatics/btr209>
4. Wang L, Park HJ, Dasari S, Wang S, Kocher J-P, Li W. CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model. *Nucleic Acids Res.* 2013;41:e74. <https://doi.org/10.1093/nar/gkt006>
5. Sun L, Luo H, Bu D, Zhao G, Yu K, Zhang C, et al. Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. *Nucleic Acids Res.* 2013;41:e166. <https://doi.org/10.1093/nar/gkt646>
6. Sun L, Liu H, Zhang L, Meng J. IncRScan-SVM: A Tool for Predicting Long Non-Coding RNAs Using Support Vector Machine. *PLOS ONE. Public Library of Science;* 2015;10:e0139654. <https://doi.org/10.1371/journal.pone.0139654>
7. Hu L, Xu Z, Hu B, Lu ZJ. COME: a robust coding potential calculation tool for lncRNA identification and characterization based on multiple features. *Nucleic Acids Res.* 2017;45:e2. <https://doi.org/10.1093/nar/gkw798>
8. Wucher V, Legeai F, Hédan B, Rizk G, Lagoutte L, Leeb T, et al. FEELnc: a tool for long non-coding RNA annotation and its application to the dog transcriptome. *Nucleic Acids Res.* 2017;45:e57. <https://doi.org/10.1093/nar/gkw1306>
9. Kang Y-J, Yang D-C, Kong L, Hou M, Meng Y-Q, Wei L, et al. CPC2: a fast and accurate coding potential calculator based on sequence intrinsic features. *Nucleic Acids Res.* 2017;45:W12–6. <https://doi.org/10.1093/nar/gkx428>
10. Singh U, Khemka N, Rajkumar MS, Garg R, Jain M. PLncPRO for prediction of long non-coding RNAs (lncRNAs) in plants and its application for discovery of abiotic stress-responsive lncRNAs in rice and chickpea. *Nucleic Acids Res.* 2017;45:e183. <https://doi.org/10.1093/nar/gkx866>
11. Schneider HW, Raiol T, Brigido MM, Walter MEMT, Stadler PF. A Support Vector Machine based method to distinguish long non-coding RNAs from protein coding transcripts. *BMC Genomics.* 2017;18:804. <https://doi.org/10.1186/s12864-017-4178-4>
12. Hill ST, Kuintzle R, Teegarden A, Merrill E, Danaee P, Hendrix DA. A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential. *Nucleic Acids Res.* 2018;46:8105–13. <https://doi.org/10.1093/nar/gky567>

13. Baek J, Lee B, Kwon S, Yoon S. LncRNA-net: long non-coding RNA identification using deep learning. *Bioinformatics*. 2018;34:3889–97. <https://doi.org/10.1093/bioinformatics/bty418>
14. Yang C, Yang L, Zhou M, Xie H, Zhang C, Wang MD, et al. LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning. *Bioinformatics*. 2018;34:3825–34. <https://doi.org/10.1093/bioinformatics/bty428>
15. Tong X, Liu S. CPPred: coding potential prediction based on the global description of RNA sequence. *Nucleic Acids Res*. 2019;47:e43. <https://doi.org/10.1093/nar/gkz087>
16. Liu S, Zhao X, Zhang G, Li W, Liu F, Liu S, et al. PredLnc-GFStack: A Global Sequence Feature Based on a Stacked Ensemble Learning Method for Predicting lncRNAs from Transcripts. *Genes*. 2019;10:672. <https://doi.org/10.3390/genes10090672>
17. Han S, Liang Y, Ma Q, Xu Y, Zhang Y, Du W, et al. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information and physicochemical property. *Brief Bioinform*. 2019;20:2009–27. <https://doi.org/10.1093/bib/bby065>
18. Camargo AP, Sourkov V, Pereira GAG, Carazzolle MF. RNAsamba: neural network-based assessment of the protein-coding potential of RNA sequences. *NAR Genomics Bioinforma*. 2020;2:lqz024. <https://doi.org/10.1093/nargab/lqz024>
19. Li M, Liang C. LncDC: a machine learning-based tool for long non-coding RNA detection from RNA-Seq data. *Sci Rep. Nature Publishing Group*; 2022;12:19083. <https://doi.org/10.1038/s41598-022-22082-7>
20. Romeijn L, Cats D, Wolstencroft K, Mei H. LncRNA-BERT: An RNA Language Model for Classifying Coding and Long Non-Coding RNA [Internet]. *bioRxiv*; 2025 [cited 2025 July 9]. p. 2025.01.09.632168. <https://doi.org/10.1101/2025.01.09.632168>
21. Goldfarb T, Kodali VK, Pujar S, Brover V, Robbertse B, Farrell CM, et al. NCBI RefSeq: reference sequence standards through 25 years of curation and annotation. *Nucleic Acids Res*. 2024;53:D243–57. <https://doi.org/10.1093/nar/gkae1038>
22. Derrien T, Johnson R, Bussotti G, Tanzer A, Djebali S, Tilgner H, et al. The GENCODE v7 catalog of human long noncoding RNAs: Analysis of their gene structure, evolution, and expression. *Genome Res. Cold Spring Harbor Lab*; 2012;22:1775–89. <https://doi.org/10.1101/gr.132159.111>
23. Romeijn L, Cats D, Wolstencroft K, Mei H. LncRNA-BERT: an RNA language model for classifying coding and long non-coding RNA. *Connect Sci. Taylor & Francis*; 2026;38:2624131. <https://doi.org/10.1080/09540091.2026.2624131>
24. Mohanty SK, Chiaromonte F, Makova KD. Evolutionary dynamics of predicted G-quadruplexes in human and other great apes. *Genome Biol*. 2025;26:161. <https://doi.org/10.1186/s13059-025-03635-1>
25. Hon J, Martinek T, Zendulka J, Lexa M. pqsfinder: an exhaustive and imperfection-tolerant search tool for potential quadruplex-forming sequences in R. *Bioinformatics*. 2017;33:3373–9. <https://doi.org/10.1093/bioinformatics/btx413>

26. Bedrat A, Lacroix L, Mergny J-L. Re-evaluation of G-quadruplex propensity with G4Hunter. *Nucleic Acids Res.* 2016;44:1746–59. <https://doi.org/10.1093/nar/gkw006>
27. Turner M, Danino YM, Barshai M, Yacovzada NS, Cohen Y, Olender T, et al. rG4detector, a novel RNA G-quadruplex predictor, uncovers their impact on stress granule formation. *Nucleic Acids Res.* 2022;50:11426–41. <https://doi.org/10.1093/nar/gkac950>