Software Evaluation for *de novo* Detection of Transposons

Matias Rodriguez and Wojciech Makałowski*

Institute of Bioinformatics, Faculty of Medicine, University of Münster, 48149 Münster, Germany

*Corresponding author: wojmak@uni-muenster.de

Abstract

Transposable elements (TEs) are major genomic components in most eukaryotic genomes and play an important role in genome evolution. However, despite their relevance the identification of TEs is not an easy task and a number of tools were developed to tackle this problem. To better understand how they perform, we tested several widely used tools for *de novo* TE detection and compared their performance on both simulated data and well curated genomic sequences. The results will be helpful for identifying common issues associated with TE-annotation and for evaluating how comparable are the results obtained with different tools.

Introduction

The vast majority of eukaryotic genomes contain a high number of repetitive DNA sequences. These sequences can be broadly classified as tandem repeats or interspersed repeats. Tandem repeats are short sequences with a length up to a few dozen bases that lie adjacent one to another and are approximate copies of the same pattern of nucleotides. Similarly, interspersed repeats are homologous DNA sequences that can be found in multiple copies scattered throughout a genome and their lengths can vary immensely from a hundred nucleotides up to more than twenty-thousand nucleotides. Most of these interspersed repetitive sequences found in genomes originated from the proliferation of transposable elements.

Transposable elements (TEs) are mobile genetic sequences possibly related to viral components that have evolved the ability to increase their abundance in a genome by making copies of themselves. The fraction of TEs in a genome can vary widely and can represent more than 80% of plant genomes [1]. To put into perspective how common they are, if we consider a well-studied case, such as the human genome, the annotated protein-coding genes represent only a very small fraction of approximately 5% of all the sequences, meanwhile TEs can make up to about 68% of the sequences [2].

Genomes and TEs have coevolved similarly to a host-parasite relationship and this led the genomes to develop multiple mechanisms to suppress TE activity as they can compromise the integrity of the genome and can cause deleterious mutations. Consequently, there is a constant evolutionary arms race between transposon activity and the host genome trying to suppress their proliferation [3]. Despite the parasitic nature of TEs, they play a fundamental role in genome evolution, contributing to plasticity, shaping, and altering the architecture of the genome. TEs contribute to gene regulatory networks as their activity can disrupt regulatory sequences modifying gene expression by altering chromatin structure, behaving as enhancers or promoters, or, when transcribed as part of a larger transcript, creating new transcript isoforms altering splicing and mRNA stability [4]. There are multiple examples of TEs that have been domesticated and proteins derived from them which were co-opted, such as the RAG1 gene from the somatic V(D)J recombination in humans and the retrotransposons that maintain the telomeres in *Drosophila.* RNA-mediated retrotransposition of transcribed genes is also a source for gene duplications that can lead to novel traits [5].

Historically, TEs were considered useless selfish sequences and their influence on genes and genomes was often dismissed [6]. It was not until the last two decades that they started to be considered as major components of genomes and important players of genome evolution, but due to the difficulties posed by their repetitive nature their annotation and role in genetic studies still continues to be neglected [7].

The correct identification of TEs is an important step in any genome project since their repetitive nature can create difficulties during *de novo* genome assemblies, breaking the continuity of contigs as a result of the same reads mapping to multiple loci [8]. They can also hinder annotation by creating conflicts with gene prediction programs if they can be found inside a host gene, carry part of a host gene when replicating, become pseudogenes, or contain spurious ORFs.

There are multiple tools for TE-detection but there are no clearly defined pipelines or software tools that could be considered as standard, as there are no clear metrics to compare the results obtained from each software [9]. Most tools also rely on a high copy number of elements for correct identification and are usually tested in organisms that have large genomes and a high abundance of TEs.

The identification of TEs can be a real daunting and a time-consuming endeavor for the amount of data that needs to be processed and compared and the challenges inherent to their complex nature. TEs are extremely diverse, they comprise multiple classes of elements that can vary immensely in sequence, length, structure, and distribution [10]. Some TEs families found in eukaryotic genomes can be very old with a majority of inactive copies due to accumulation of mutations or fragmentation during the insertion process. This means that remains of antique copies from a family can be very divergent from active TEs, making the detection of the remnants of decayed copies or the definition of consensus sequences a real challenge that is hindered by the great variability of TEs within the same family. The proliferation of TEs can also result in the generation of nested TEs and some families show a clear preference for jumping into other TEs that act as hotspots for insertion [11], making the detection and correct annotation of them even more difficult.

There are well curated TE databases, such as RepBase [12] or Dfam [13], with libraries of consensus sequences. A homology-based approach relies on the TE sequences from these libraries which are then mapped against the studied genome. To identify new TEs a *de novo* approach is used and there are abundant software alternatives which rely on different strategies ranging from structural information, periodicity, k-mer counting, or repetitiveness, among others [14]. When a new species is sequenced, a strategy which uses only information from curated databases is not enough and it is necessary to use a *de novo* strategy to identify novel families and species-specific TEs.

In this work we compare TE detection software which are widely used by researchers and we assess their performances on genomes with well curated TE annotations. We ran a number of *de novo* TE detection software packages on simulated sequences and genome sequences and then compared and evaluated their performance in detecting a wide variety of divergent TE families. A particular scenario that we tried to consider is the detection of transposons in smaller genomes of around a hundred million bases. In all cases the software for identification of TE rely on the presence of a large number of the same family of elements and that is usually not common in smaller genomes where a lower number of copies of TEs is expected.

Methods

Datasets

Genomic data with annotated TEs were downloaded from the UCSC Genome Browser database [15]. The TE annotation provided by UCSC Genomes was obtained from mapping TEs from the RepBase database [12] against each genome using RepeatMasker [16]. The sequence data sets we used varied from 46.7 Mb for the human chromosome 21 and 137.5 Mb for the zebrafish chromosome 1 (see Table 1).

Table 1. Datasets used for testing the TE *de novo* detection tools.

Dataset	Sequences	Length	TE-fraction
Human genome	Chromosome 21	46.7 Mb	42.92%
Zebrafish	Chromosome 1	137.5 Mb	56.81%
Fruit fly	Whole genome	137.6 Mb	17.65%
Simulated data	Simulation	100.1 Mb	60.00%

Simulated data

We used a Python script to simulate an ideal scenario where the composition, coordinates, and divergence of all the TEs are already known. This script takes input from a configuration file for GC content, TE sequences, number of copies, expected divergence (mutations and indels), percent of fragments, and nesting. The script starts by simulating a random sequence of a predefined length and a GC content that constitutes the base sequence where TEs are going to be inserted. Then it obtains the name of each TE and the number of copies from the configuration file and random positions are chosen and assigned to each TE. In the next step, TE sequences are loaded from a library and the information about divergence and fragmentation is taken into account to generate random mutations and fragments that are inserted into the base sequences. The last step takes all of this information to generate a fasta file with the whole new sequence with TEs and a GFF file with all the coordinates and relevant information. For the simulated dataset we used a base sequence of 40 MB and inserted 60 MB from 20 different families downloaded from the Dfam database [13].

Software

In this work we compared strategies for *de novo* detection of TEs using k-mer based tools and self-comparison tools. We tested three k-mer counting tools: Red, P-Clouds, and phRaider. Due to the nature of the algorithms employed by k-mer counting software, these tools are extremely fast and usually don't require much computational power. Nevertheless, they usually require a big amount of RAM to store data structures, so they may not scale up well with large genomes. Red identifies candidate repetitive regions giving them a score, then processes these results using signal processing and the second derivative. These filtered data are used to train a Hidden Markov Model that scans the genome for candidate TEs. As described by the author, it is a novel repeat discovery system that trains itself automatically on an input genome [17]. P-Clouds counts oligonucleotides, then arranges them into clusters of “probability clouds” that are related oligonucleotides that occur as a group more often than expected by chance. Then it annotates the genome by finding stretches with a high density of oligos present in these “probability clouds” [18]. The premise of phRAIDER is to use spaced seeds to specify match patterns, i.e., to permit the search of substrings allowing mismatches in certain positions. Then it scans the genome searching for highly frequent seeds and how they overlap [19]. The limitation of these tools is the fact that they don't make any attempt to classify found repeats. Moreover, there is no information provided on relation between the detected individual elements.

We also compared three self-comparison tools: RepeatScout, REPET (TEdenovo pipeline), and RepeatModeler. RepeatScout uses high frequency seeds and extends each seed to a progressively longer consensus sequence, following the dynamically inferred alignments between the consensus sequence and its occurrences in the genome. The alignment score encourages boundaries shared by some but not necessarily all alignments; it uses a standard SW-algorithm to extend until n-iterations fail to improve the score [20]. REPET is a package consisting of two pipelines, one for detection of TE: TEdenovo, and another for their annotation: TEannot [21, 22]. Both of these pipelines are fully configurable and each step can be parametrized. The TEdenovo pipeline by default starts self-comparison of the input genome with BLASTER, a modified version of BLAST. Then it clusters the high scoring pairs using three tools: RECON, GROUPER, and PILER, grouping closely related TE sequences. Finally it performs a multiple alignment using MAFFT or MAP with the aim of having a consensus sequence for each TE family. Here, we are interested in the ability of the software to detect TEs, so we used only the TEdenovo pipeline. Finally, RepeatModeler is a pipeline that uses as an input the outputs of three other software, namely RECON, RepeatScout, and Tandem Repeats Finder. Additionally, it uses LTRHarvest or LTRretriever for LTR-TE detection [23]. RepeatScout, RepeatModeler, and REPET all give as a final result a fasta file with a consensus sequence for each type of TEs they could identify. Afterwords, we need to map back these consensus sequences to the genome to identify individual copies of TEs and get their coordinates. For this step we used the popular tool RepeatMasker, version 4.0.9 [16], using the three tools' outputs as libraries to screen the genomes for TEs.

Another tool used for detecting simple repeats is Tandem Repeat Finder (TRF) [24], a software which models tandem repeats using a probabilistic model. We used it to filter out simple repeats obtained by the k-mer counting tools and also to assess the ability of self-comparison software to cope with simple sequence repeats. TRF was run with default parameters on all the sequences analyzed and the results obtained were merged when an adjacent or overlapping annotation was reported. These results were converted into a GFF file for easing further analysis.

Pipeline

All the software were tested with default parameters as we intended to compare the average performance of each tool without tuning their optional parameters (see Figure 1). K-mer counting methods are expected to find all the high frequency k-mer, including simple repeats and interspersed repeats. For k-mer counting software after getting the results we ran Tandem Repeat Finder (TRF) with default parameters to filter out tandem repeats. The self-comparison tools usually include some steps for filtering simple repeats and frequently make use of TRF, so this step was not replicated with these software.

The type of results produced by each set of tools is also quite different due to the different strategies used. K-mer counting tools return the coordinates in the genome with the regions where high frequency k-mers were found, meanwhile self-comparison software returns the consensus sequences of the TE candidates found as a fasta file. So for the latter set of tools we mapped back the consensus against the original sequences using RepeatMasker, running it with default parameters. The results obtained from all the different tools were transformed to GFF format for further processing.

Then GFF files were sorted by coordinates and immediately adjacent, overlapped, or internal coordinates were merged as one, as the main idea is only the identification of transposon sequences. This step is necessary particularly in k-mer counting software which have the tendency to annotate many overlapping and fragmented repeats. For all the datasets tested we have as a reference the annotations downloaded from the UCSC Genomes database.

TE-models' comparison

TE-models generated by self-comparison software on simulated data were compared to original TE-families using blastn with default paramters.

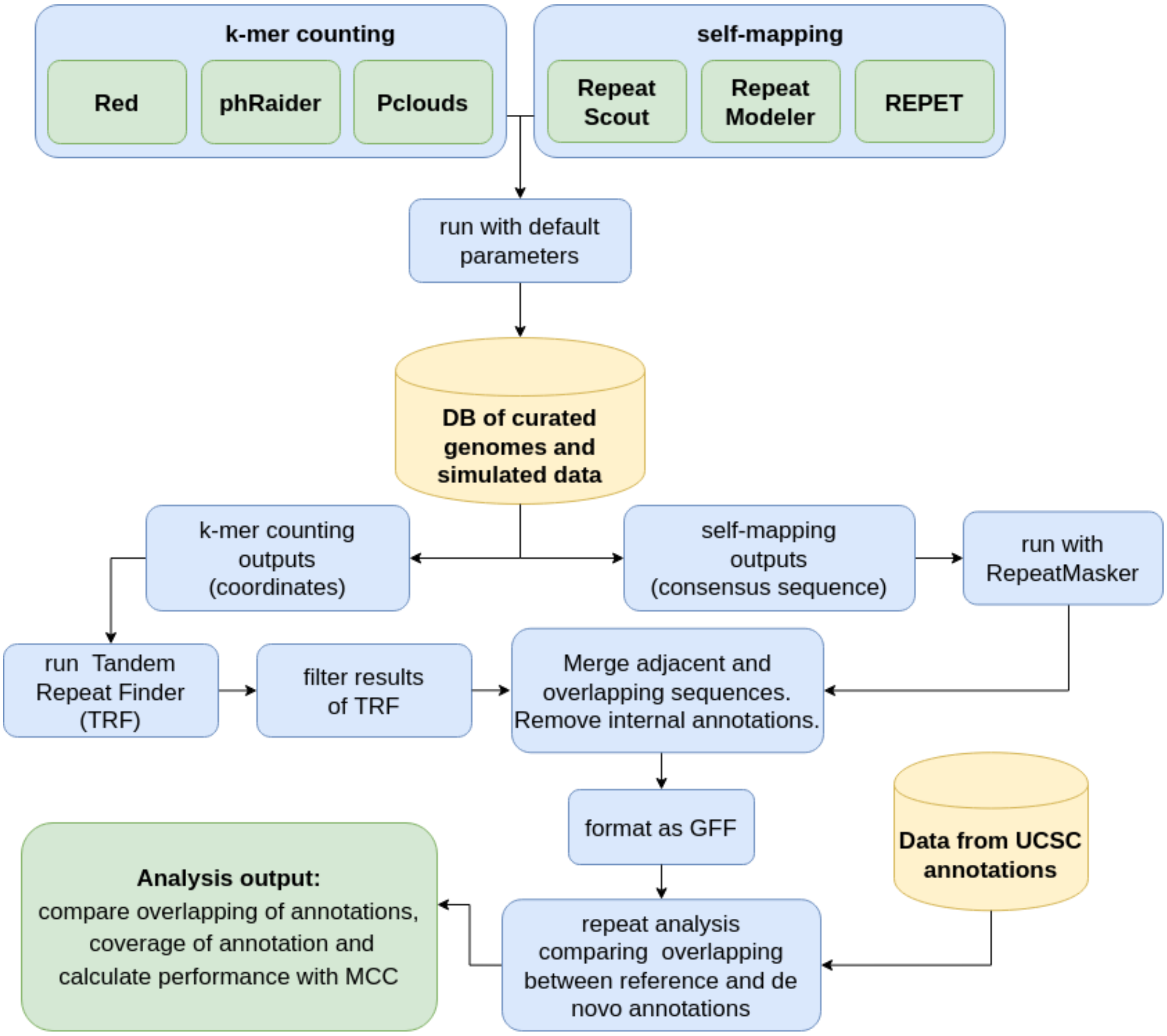
k-mer counting
Red
phRaider
Pclouds
self-mapping
Repeat Scout
Repeat Modeler
REPET
run with default parameters
DB of curated genomes and simulated data
k-mer counting outputs (coordinates)
self-mapping outputs (consensus sequence)
run with RepeatMasker
run Tandem Repeat Finder (TRF)
filter results of TRF
Merge adjacent and overlapping sequences. Remove internal annotations.
format as GFF
Data from UCSC annotations
Analysis output: compare overlapping of annotations, coverage of annotation and calculate performance with MCC
repeat analysis comparing overlapping between reference and de novo annotations

Figure 1. Pipeline used for testing and comparing the performance of *de novo* detection tools.

Analysis

With the results of TE detection obtained from each software we created GFF files that were then compared against the original files from UCSC database with RepeatMasker mapping results. A custom Python script was used to obtain the overlapping regions of two GFF files and where there are differences in the annotation, it allowed us to compare the coordinates of the reference and the ones obtained by the TE *de novo* software. This way of evaluating the data is useful in order to create a confusion matrix that can be used as input for a binary classifier test that allows us to compare the performance of different software against a reference. When the reference annotation and the new annotation agree on the coordinates, these bases are counted as true positives, or if nothing is annotated in both, these bases count as true negatives (Figure 2). If the new annotation has bases not covered by the reference annotation, we consider them as false positives, and similarly if annotations in the reference are missing in the new annotation, these are counted as false negatives (Figure 2).

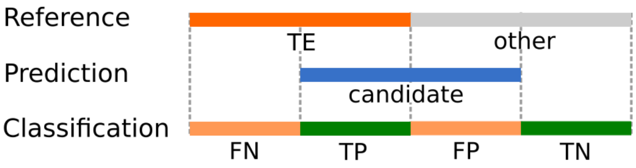

Figure 2. Classification of the results obtained after comparing the reference annotation and the predicted TEs. We have TEs in the reference genome annotation and TE candidates as a prediction. Then comparing both they can be classified as false negatives (FN), true positives (TP), false positives (FP), or true negatives (TN).

With this kind of data we have a binary classification problem, where each category can be classified using a confusion matrix. There are multiple tests to evaluate and compare the results obtained by a binary classifier which make use of a confusion matrix data and one of the most commonly used methods is the Matthews Correlation Coefficient (MCC). The MCC has the advantage that it uses all four quadrants of a confusion matrix considering the proportions of each class and requires that in both classes negative and positive elements are correctly classified, performing well even when using imbalanced data and when one class is underrepresented [25]. The MCC evaluates the results obtained from a prediction, as in this case the TE *de novo* software TE candidates, against the known annotated data. The values of MCC range from -1 to +1, where a value of -1 is obtained when all the predictions are wrong, 0 when results are not better than random guessing, and 1 where all predictions are correct. In this work we used the MCC as a measure of the performance obtained from the different software tools. Additionally, we developed several R scripts for plotting GFF coordinates which visually compare the annotations obtained from each tool.

Results

As mentioned above, the two groups of programs provide different types of results. While k-mer counting software provided a list of regions that are occupied by repetitive sequences, self-comparison software analyzed here returned sets of repeats' models. These models can be next used to scan a genome and annotate individual repeats, including TEs. For this step, we used a popular program, namely RepeatMasker (see Methods section).

Model building

Three self-comparison based programs were used to create TE-models for both real genomic data and simulated sequences. The results of the latter are the most informative as we knew the exact number of expected TE-families. Interestingly, all three programs generated more TE-models than we used for the simulation (see Table 2). REPET created the highest number of models even though it failed to report a model based on 730 Alu insertions in the simulated sequence.

Software	Number of models generated
RepeatScout	30
RepeatModeler	80
REPET	82

Table 2. Number of TE models generated by each software from a simulated sequence containing TEs of twenty different families. Please note that REPET failed to generate an Alu model.

RepeatScout generated the smallest number of TE-models (30) and only in few cases more than two models for a given TE-family: three for L1 and four for Polinton. However, it has a tendency to create homo-dimeric elements, for instance Copia, DIRS, HERVL (see Table S1). On the other extremum lies REPET. It not only generated the highest number of models but some of them were dimeric and hybrid. The latter were caused by a few nested repeats, for instance a Jockey nested in a Ngaro or a Tc1-Mariner nested in a HERVL. In general, longer elements tend to give rise to several models by both RepeatModeler and REPET. For instance, 5.5 kb long L1 element is a source of six models in RepeatModeler analysis and nine models in the case of REPET. Polinton, which is 18.5 kb, resulted in eight models in both RepeatModeler and REPET and four models in the case of RepeatScout (see Table S1). Interestingly, some of these models overlap each other, suggesting that they could be merged during manual curation.

In our simulation, we “mutated” individual TE-copies up to twenty percent divergence from the reference sequence and many of the individual copies were truncated at the 5’ end. In general, a consensus sequence recovery at the nucleotide level was very good, with the average sequence identity of models to their respective reference sequences at 97.3% (stdev = 3.76). However, many of TEs were broken by a given software into several models. Probably the best example is Polinton, based on which RepeatModeler and REPET created nine models each and none of these covered the whole Polinton sequence that was used in the simulation (see Figure 4). The shortest transposon inserted into the simulated data was the 311 nucleotide long AluY element. The individual sequences were "mutated" to average 13% divergence from the reference sequence and 67 of them were truncated at their 5' end up to 30% of the sequence length. RepeatScout performed the best, returning a 208 nt long model with the sequence identical to the reference and just 3 nt missing from the 5' end. Surprisingly, REPET didn't report any models based on these sequences. Finally, RepeatModeler created three different models: one almost ideal with just a 5' terminal guanosine missing and two others, a bit shorter but with extra eight and thirty-five nucleotides added to their 5' end (see Table S1).

When analyzing the simulated data, one trend became clear, namely that on occasions multiple models are generated for the same TE and there are common patterns observed for each software. A characteristic of RepeatModeler is that it tends to generate redundant models, with up to six to eight models for the longest TEs. This is also a common behavior observed with REPET, where many fragments were generated. Another interesting observation that only occurs with REPET is that in some models part of a nested TE was included into a model resulting in chimeric models. With RepeatScout there is much less redundancy with the number of models, but again something unique happens and some of the reported models are total or partial duplicates of the original TE.

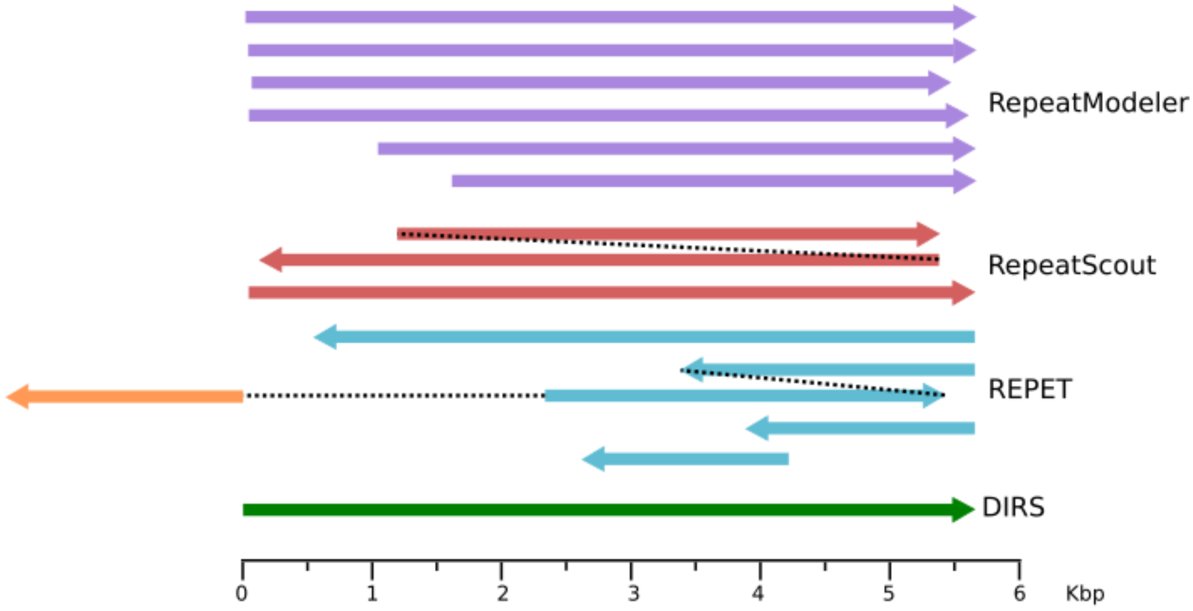

Figure 3. Consensus sequences generated by each software for a single TE from the DIRS family. The length and position reflect the mapping to the TE, black dotted lines show the continuation of the same model, the orange segment represents a fragment coming from another TE (TRANSIB).

An example of different models generated for a DIRS TE of 5.6 kb which were particularly difficult to resolve is shown in Figure 3. This particular TE RepeatModeler generated six different models of different lengths. These models are on average fifteen percent diverged at the sequence level. There were two models calculated by RepeatScout, one almost identical to the reference and another one almost twice the length of the original and consisting of a duplication leading to erroneous homodimer. Interestingly, the two copies of this homodimer are complementary to each other as they lie on opposite strands compared to the reference sequence. REPET reported four models in total. Two are relatively short, encompassing about one-third of the reference sequence and partially overlapping in a head-to-tail orientation. Another model is a hybrid TE consisting of two overlapping fragments of DIRS element in a head-to-tail orientation with a fragment of TRANSIB transposon. The fourth model closely resembles the original TE but is truncated by about 240 nucleotides at its 5' end (see Figure 3).

In Figure 4 we present models generated for another TE, namely Polinton-1_DR [26]. The full-length transposon is 18.5 kb, including 350 bp terminal inverted repeats. All three software compared in our study reported few models for this TE but none of the models recovered the full length TE (see Figure 4 and Table S1). Interestingly, both REPET and RepeatModeler generated similar close-to-full-length models that are missing one of the inverted repeats, while RepeatScout's longest model misses both inverted repeats. However, inverted repeats were reported as separate models by each of the program.

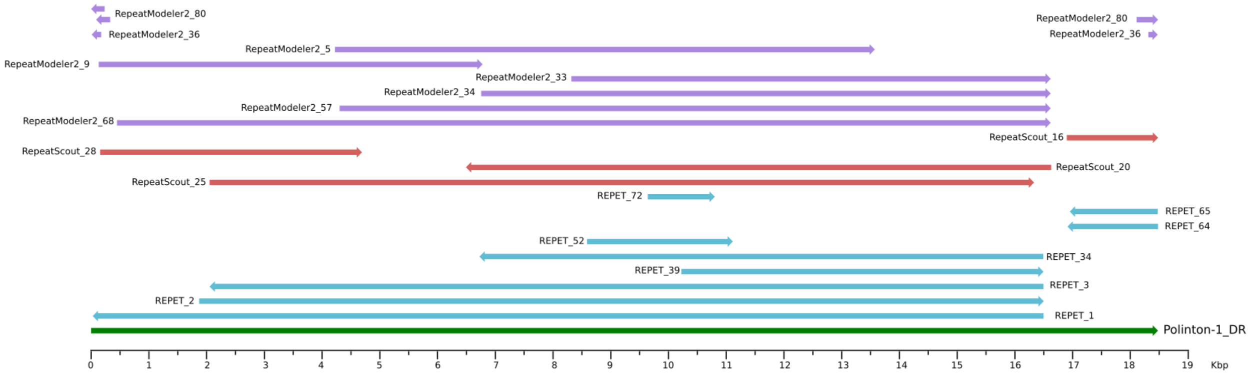

Figure 4. Different models created for Polinton-1_DR transposon aligned against Dfam model DF0002823.2.

In the real data from model organisms, RepeatScout created the highest number of models with almost three-thousand consensus for zebrafish chromosome 1 (see Table 3). This is in contrast to the simulated data where RepeatScout generated the least number of models. REPET lies on the other extreme of the spectrum with just 65 TE models for human chromosome 21. The smaller number of models generated for human data might be linked to the smaller sequence data compared to the two other datasets. However, based on the TE-annotation, the total length of TEs in the fruit fly genome is comparable to the total length of TEs in human chromosome 21, 24 MB versus 20 Mb, respectively. In general, the real data seem to harbor more versatile repertoire of TEs than our simulated data resulting in many more TE models (compare Table 2 and Table 3). However, it is quite difficult to realize any general pattern of TE-discovery using these programs, simply there is none.

Table 3. Number of models generated by three software in real sequence analysis.

Software	Number of models generated		
	Zebrafish chromosome 1	Fruit fly genome	Human chromosome 21
RepeatScout	2919	2593	464
RepeatModeler	1779	686	428
REPET	342	557	65

Based on our simulated data analysis it is clear that none of the analyzed software is able to compute a repeat consensus sequence accurately. While the sequence of a repeat can be calculated with confidence, the structure of the repeat should be inspected manually and edited accordingly. This is especially important for longer transposons.

Individual repeats annotation

To get a better idea of the different results of the *de novo* annotation obtained by the six tools used, we plotted the coordinates of each one in tracks along with the reference annotation, as shown in Figure 5. Simply by visualizing the results it is quite evident that there is a tendency to get a fragmented annotation when using k-mer counting tools, particularly Pclouds and phRaider. Red uses a smoothing function to merge nearby high frequency k-mers, giving less fragmented results, as shown in Figure 5 and 6. As it was expected due to the methodology used, the best results for detecting transposons were obtained by self-comparison software, but as it is shown more in detail in Figure 6, most of the predictions are fragmented TEs, annotations without clear borders, or missing some smaller or incomplete elements.

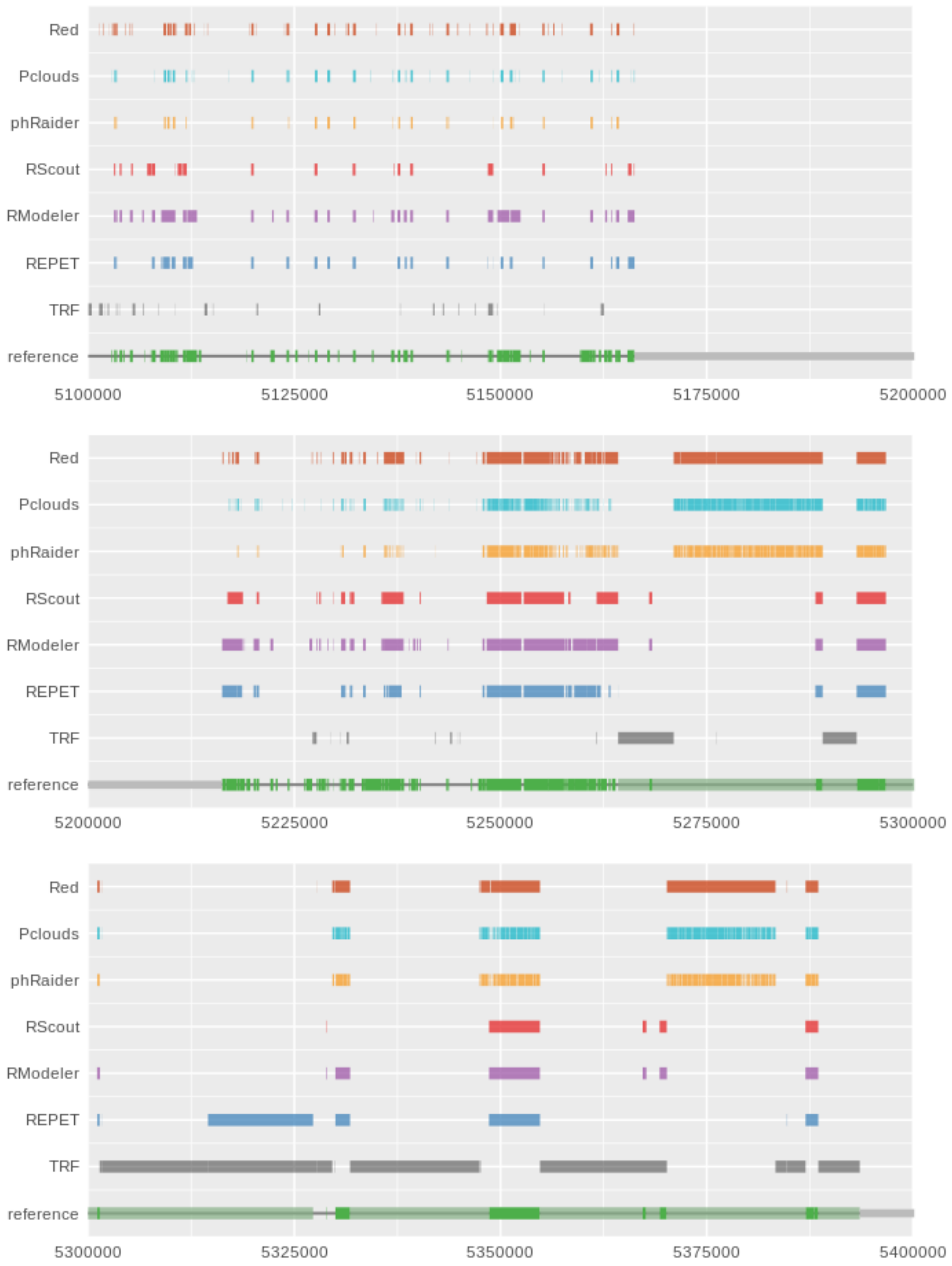

Figure 5. Different tracks of the coordinates obtained from a *de novo* identification of transposons using six different software tools for detecting interspersed repeats. In the reference track, green blocks are transposons.

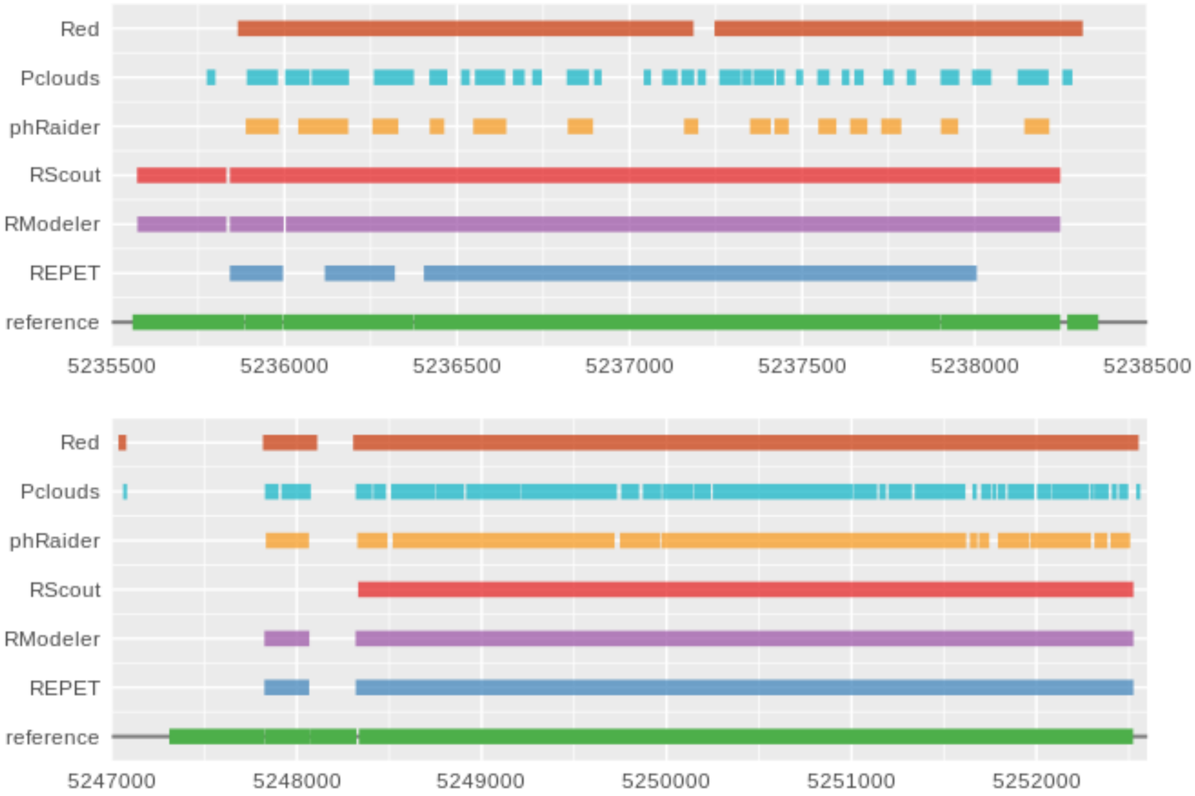

Figure 6. Comparison of the fragmentation of results in a region of the human chromosome 21. In each track are the predictions obtained from each tool and in green the reference. Notice how most of the results are usually incomplete or fragmented.

For self-comparison software, in most cases RepeatModeler got the best results, but RepeatScout obtained also comparable results. REPET failed in some scenarios and particularly with short divergent fragments, one of the reasons could be that this software was developed with the idea to be used with large genomes and here all the tests were run with sequences of around 100 MB. We compared the annotation results obtained using simulated sequences with known identities between TEs ranging from 60% to 100% and then compared the coverage of the annotation in relation to it, as is shown in Figure 5. It is expected that TEs with higher identities are detected more precisely. Indeed TEs with a higher identity are better detected by all software and the differences seen are inherent to the performance of each tool. For k-mer counting software, Red performs significantly better than the rest, e.g at 70% identity, Red detects approximately 85% of the TE regions. Meanwhile, Pclouds detects about 60% and phRaider 25% (Figure 7). The self-comparison software display a much better and uniform performance and are less affected by more divergent TEs (Figure 7).

When we also consider the different TE orders annotated and the proportion of coverage for the TEs of each order, we observe that there's no significant difference between them and all the software have a consistent performance (supplementary material, Figure S1).

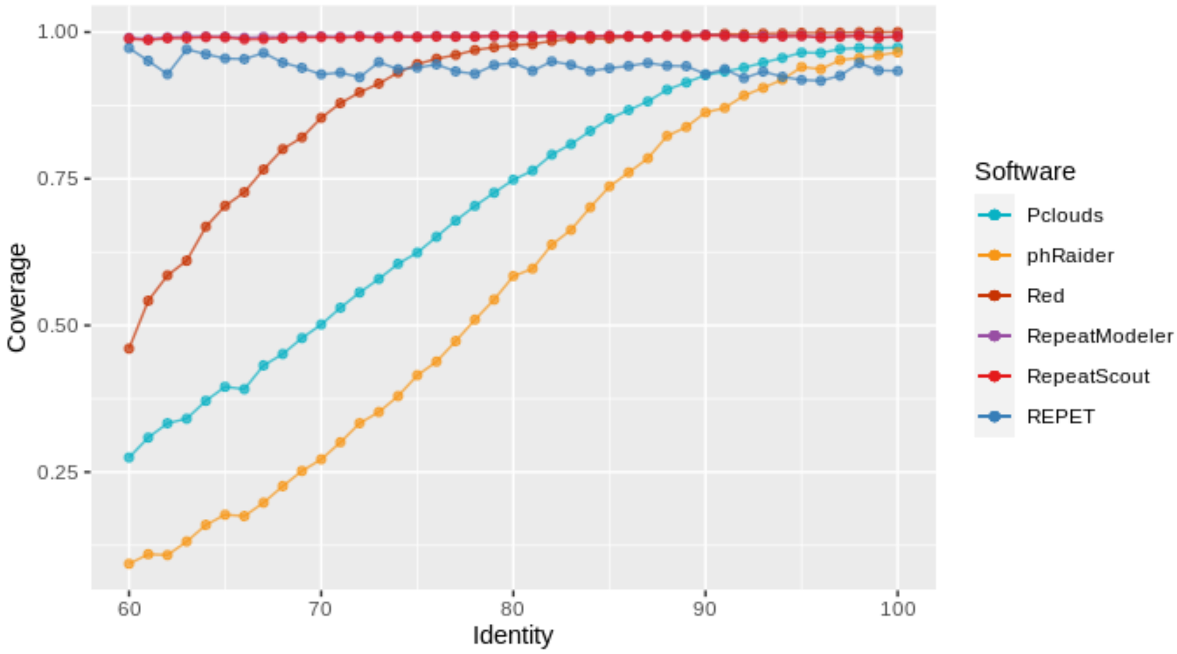

Figure 7. Coverage of TEs in simulated sequence in relation to their average identity. In k-mer counting software there's a great drop in the detection of more divergent TEs, this behavior is not seen when using self-comparison tools. RepeatModeler and RepeatScout results are almost identical and they are overlapped in this plot.

Finally we evaluated the performance of each tool against the datasets using the Matthews Correlation Coefficient (Figure 8). To evaluate the results it is important to consider not only the raw performance of each tool but also the difficulty to run, configurability, and speed. K-mer counting software usually only accept a few parameters such as k-mer length, minimum frequency, and length; but these tools are usually very easy to run and require little computing power while being incredibly fast. However, one of the performance downsides can be the requirement to store large data structures in memory. In this category, with defaults parameters Red outperformed pClouds and phRaider and this can be explained by the fact that Red merges nearby k-mers more frequently than the others, giving less fragmented results. Self-comparison software employ a strategy that requires much more computational resources. They are also more time consuming and can be more complex to install, configure, and run. In our tests RepeatModeler obtained the best results, but it is interesting to notice that RepeatScout is part of RepeatModeler and also quite fast compared to the latter also obtained good results. REPET on the other hand was probably not the best tool for this type of analysis. REPET has a default configuration but also different tools can be added to the pipeline and each step is highly configurable, but one of the downsides is that it can be very complex to configure and run for an unexperienced user.

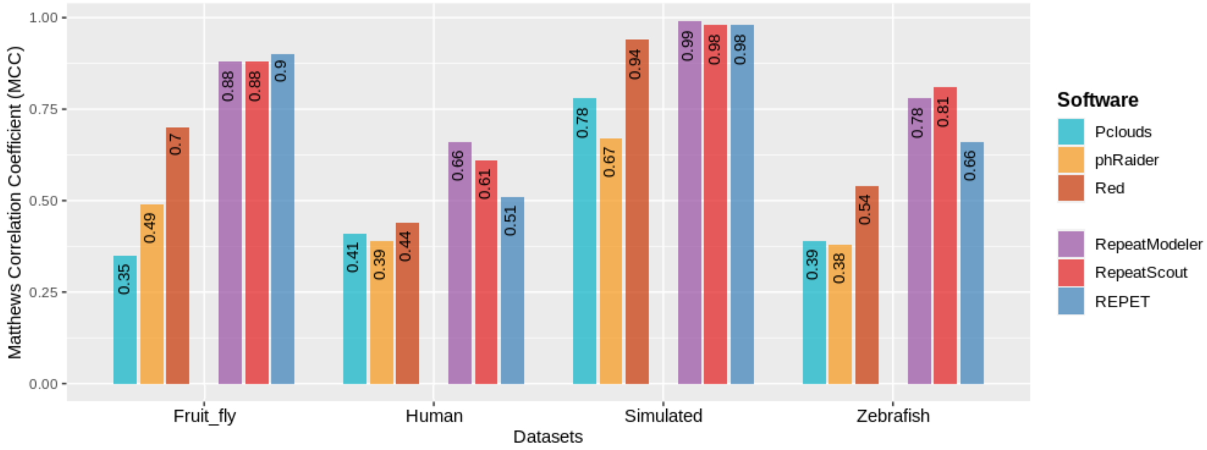

Figure 8. Matthews Correlation Coefficient values showing the performance of each tool tested with the datasets tested.

In general, for a fast assessment of interspersed repeats, Red can be useful, acknowledging of course its limitations when it comes to low complexity sequences. For more in depth studies with small genomes, RepeatModeler seems to be the best option. It is also interesting to note that RepeatScout has a really good performance if we take into consideration speed and computational requirements.

Nevertheless, for *de novo* detection of TEs, extensive manual curation and using multiple tools for confirmation of the results obtained is necessary. We also found that MCC can be used as a fast and reliable test to compare the performance of these software and can give a general idea of which tool is best suited for each task.

Conclusions

We tested a number of tools for *de novo* detection of TEs. The results were compared using the MCC against a reference of annotated TEs. As expected, self-comparison tools performed better than k-mer counting ones, with RepeatModeler beating competitors in most datasets. However, even for RepeatModeler, the results are far from satisfactory based on the reference annotation. There is a tendency for most tools to identify TE-regions in a fragmented manner and it is also frequent that small TEs or fragmented TEs are not detected. We recognize that some of the results obtained may be improved by fine tuning of parameters; some tools like REPET are fully customizable and more tools can be added to the pipeline, although this can be challenging for an average user. In conclusion, most of the contemporary tools for *de novo* detection of TEs are far from being perfect and the identification of TEs is still a challenging endeavor as it requires a significant manual curation by an experienced expert.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

Scripts and simulated sequences used for the software evaluation are deposited at https://github.com/IOB-Muenster/denovoTE-eval.

Funding

This work was partially funded by the DAAD Research Grants - Doctoral Programmes in Germany, 2018/19 (57381412) to MR and internal funds of Institute of Bioinformatics.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

WM initiated and designed the project. MR executed the project and wrote a draft manuscript. Both authors worked on the final manuscript.

Acknowledgements

Not applicable.

Bibliography

1. Schnable PS, Ware D, Fulton RS, Stein JC, Wei F, Pasternak S, et al. The B73 maize genome: complexity, diversity, and dynamics. Science. 2009;326(5956):1112-5.

2. de Koning APJ, Gu WJ, Castoe TA, Batzer MA, Pollock DD. Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. Plos Genet. 2011;7(12).

3. Jurka J, Kapitonov VV, Kohany O, Jurka MV. Repetitive sequences in complex genomes: Structure and evolution. Annu Rev Genom Hum G. 2007;8:241-59.

4. Makalowski W. Genomic scrap yard: how genomes utilize all that junk. Gene. 2000;259(1-2):61-7.

5. Kubiak MR, Makalowska I. Protein-Coding Genes' Retrocopies and Their Functions. Viruses. 2017;9(4).

6. Ohno S. So much “junk” DNA in our genome. In: Smith H, editor. Evolution of Genetic Systems: Brookhaven Symposia in Biology. 23. New York: Gordon and Breach; 1973. p. 366-70.

7. Biemont C. A Brief History of the Status of Transposable Elements: From Junk DNA to Major Players in Evolution. Genetics. 2010;186(4):1085-93.

8. Ricker N, Qian H, Fulthorpe RR. The limitations of draft assemblies for understanding prokaryotic adaptation and evolution. Genomics. 2012;100(3):167-75.

9. Hoen DR, Hickey G, Bourque G, Casacuberta J, Cordaux R, Feschotte C, et al. A call for benchmarking transposable element annotation methods. Mobile DNA-Uk. 2015;6.

10. Wicker T, Sabot F, Hua-Van A, Bennetzen JL, Capy P, Chalhoub B, et al. A unified classification system for eukaryotic transposable elements. Nat Rev Genet. 2007;8(12):973-82.

11. Gao CH, Xiao ML, Ren XD, Hayward A, Yin JM, Wu LK, et al. Characterization and functional annotation of nested transposable elements in eukaryotic genomes. Genomics. 2012;100(4):222-30.

12. Bao WD, Kojima KK, Kohany O. Repbase Update, a database of repetitive elements in eukaryotic genomes. Mobile DNA-Uk. 2015;6.

13. Hubley R, Finn RD, Clements J, Eddy SR, Jones TA, Bao WD, et al. The Dfam database of repetitive DNA families. Nucleic Acids Res. 2016;44(D1):D81-D9.

14. Makalowski W, Gotea V, Pande A, Makalowska I. Transposable Elements: Classification, Identification, and Their Use As a Tool For Comparative Genomics. Methods Mol Biol. 2019;1910:177-207.

15. Haeussler M, Zweig AS, Tyner C, Speir ML, Rosenbloom KR, Raney BJ, et al. The UCSC Genome Browser database: 2019 update. Nucleic Acids Res. 2019;47(D1):D853-D8.

16. Smit A, Hubley R, Green P. RepeatMasker Open-4.0 2013-2015 [cited 2020 10.02]. Available from: http://www.repeatmasker.org.

17. Girgis HZ. Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale. Bmc Bioinformatics. 2015;16.

18. Gu WJ, Castoe TA, Hedges DJ, Batzer MA, Pollock DD. Identification of repeat structure in large genomes using repeat probability clouds. Anal Biochem. 2008;380(1):77-83.

19. Schaeffer CE, Figueroa ND, Liu XL, Karro JE. phRAIDER: Pattern-Hunter based Rapid Ab Initio Detection of Elementary Repeats. Bioinformatics. 2016;32(12):209-15.

20. Price AL, Jones NC, Pevzner PA. De novo identification of repeat families in large genomes. Bioinformatics. 2005;21:I351-I8.

21. Quesneville H, Bergman CM, Andrieu O, Autard D, Nouaud D, Ashburner M, et al. Combined evidence annotation of transposable elements in genome sequences. Plos Comput Biol. 2005;1(2):166-75.

22. Flutre T, Duprat E, Feuillet C, Quesneville H. Considering Transposable Element Diversification in De Novo Annotation Approaches. Plos One. 2011;6(1).

23. Flynn JM, Hubley R, Goubert C, Rosen J, Clark AG, Feschotte C, et al. RepeatModeler2 for automated genomic discovery of transposable element families. P Natl Acad Sci USA. 2020;117(17):9451-7.

24. Benson G. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res. 1999;27(2):573-80.

25. Boughorbel S, Jarray F, El-Anbari M. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. Plos One. 2017;12(6).

26. Kapitonov VV, Jurka J. Self-synthesizing DNA transposons in eukaryotes. Proc Natl Acad Sci U S A. 2006;103(12):4540-5.

Supplementary materials

Table S1. Comparison of the consensus models obtained for each single TE inserted in the simulated sequence. With * are indicated models were the models include a duplication or an extension of the same TE and with # are indicated models that are the combination of two TEs which originated from nested TEs.

TE_id	Count	Average identity	Standard deviation	Number of indels	Target site duplication	Length	Number of truncated elements	Number of nested element
AluY	730	87	12	16	y	311	67	11
BEL12-I_DR	550	89	12	10	y	7730	80	12
Copia1_DM	680	87	14	10	y	4530	65	10
DIRS-1_DR	710	85	15	12	n	5634	70	5
G5_DM-Jockey	590	91	10	15	y	4856	75	16
GYPSY8_DM	470	84	10	15	y	4955	74	12
hAT-9_DR	480	82	12	11	y	2386	65	13
Helitron-1N3_DR	550	83	10	11	n	1843	65	10
HERVL	590	88	14	14	y	6542	50	7
L1-4_DR	620	78	9	16	y	5548	72	13
Merlin1_HS	580	80	11	13	y	1175	60	18
Ngaro1_DR	560	86	12	11	n	6578	73	9
Penelope-1_DR	810	85	11	20	y	4446	60	11
piggyBac-N1_DR	610	80	10	19	y	1005	64	20
Polinton-1_DR	420	81	11	20	n	18485	70	10
RTE1	570	90	13	18	y	3291	71	11
SINE3-1	580	85	13	10	y	590	70	7
TC1-2_DM	590	83	14	12	y	1644	70	16
TRANSIB1	670	81	10	12	y	3014	68	9
Turmoil2	650	79	9	10	y	6999	69	15

Table S2. TEs inserted into simulated data. Transposon ids after Dfam.

Original TE-families		Length of the models created		
TE	**Length**	**RepeatScout**	**RepeatModeler**	**REPET**
AluY	311	308	345 320 310	-
BEL12	7730	7724	7751 7730 6413	11148* 7750 2769
Copia	4530	8843*	4532 4514 4326 4076	4652 4314
DIRS	5654	5644 9385*	5672 5653 5610 5445 4633 4062	7134# 5401 1824 1727
G5	4856	4854	4851 4843 4836 3151	7502 * 5232 5197 4904 3967 2359
GYPSY	4955	4360 3920	4938 4012 3951 3373	5068 4682 2061 1445 1256
hAT	2386	2380	2380 2380	2426
Helitron	1843	1839 1472	1832 1823 1234	1907 1906 1744 871 633
HERVL	6542	12634*	6549 6523 6094	11495# 6553 3886 3169

Original TE-families		Length of the models created		
TE	**Length**	**RepeatScout**	**RepeatModeler**	**REPET**
L1	5548	4460 3379 2415	5496 5488 3386 2588 2478 1289	5784 5605 5443 2855 2142 1794 1639 1179 667
Merlin	1175	1168	1172 1167 744	1231 1216 1089 629
Ngaro	6578	6571 6020	6568 6566 6563 6156	7798# 6652 6379 2487 2445
Penelope	4446	4437 2276	4432 4407 4189 3910 3198 3908	5077 3251 812
piggyBac	1005	997	1002 990 840	1067 1055 966 641 477
Polinton	18485	14288 10127 4536 1513	17626 13482 11066 9356 8330 6655 372 169	16912 16015 15368 10418 6631 2793 1690 1216
RTE1	3291	3287	3286 3270 3212	3343 3200 3128 1222
SINE3	590	587	587 581 578	624 559

Original TE-families		**Length of the models created**		
TE	**Length**	**RepeatScout**	**RepeatModeler**	**REPET**
TC1	1644	1634	1636 1627 1472	1689 771
TRANSIB	3014	3002	2995 2546 2378	3094 2858 1166
Turmoil	6999	6676	7016 6985 6974 6970 4201	7113 6598 5956 5834# 1757 1381 1100

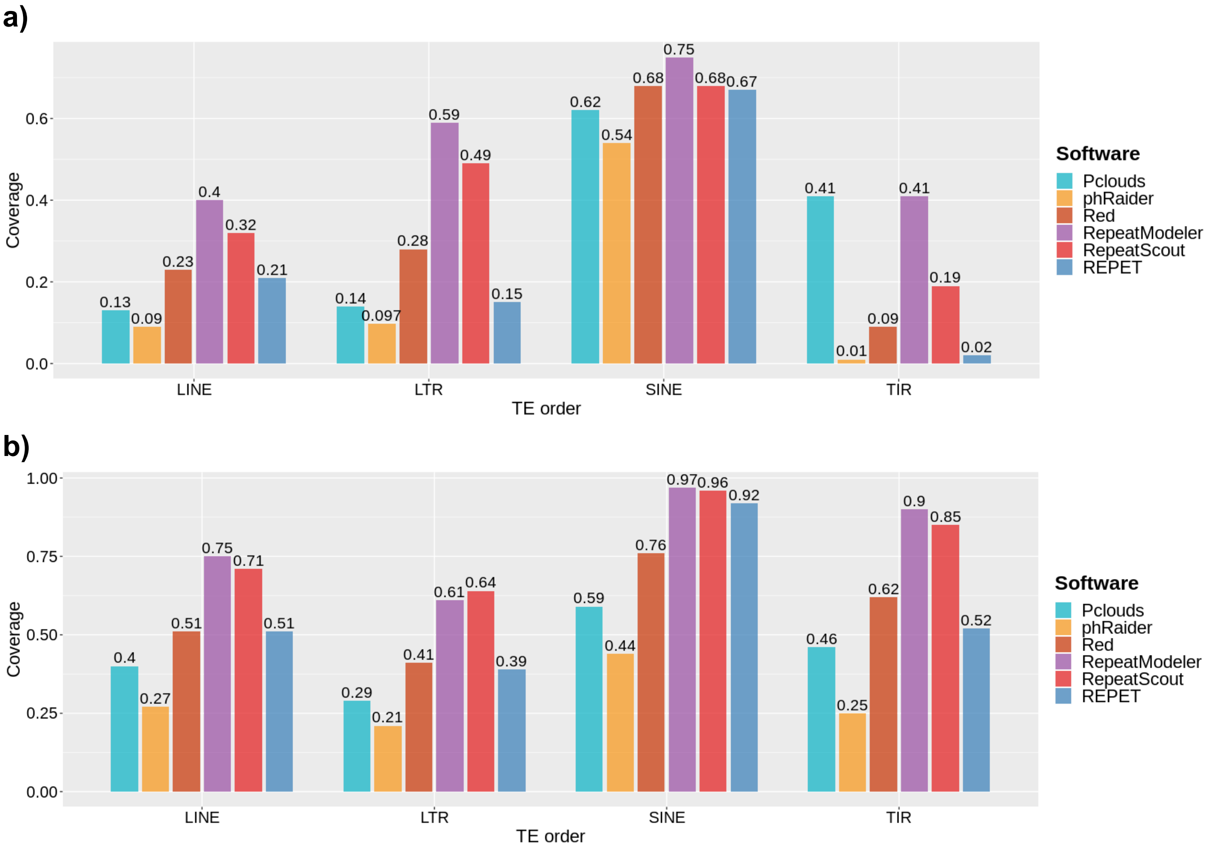

Figure S1. Coverage of each TE order in the human (a) and zebrafish (b) dataset.