Bioinformatics Advances, 2024, **00**, vbae149
https://doi.org/10.1093/bioadv/vbae149
Advance Access Publication Date: 9 October 2024
Original Article

Software

ULTRA-effective labeling of tandem repeats in genomic sequence

Daniel R. Olson[1,*] **and Travis J. Wheeler**[1,2]

[1]Department of Computer Science, University of Montana, Missoula, MT 59812, United States
[2]Department of Pharmacy Practice & Science, R. Ken Coit College of Pharmacy, University of Arizona, Tucson, AZ 85721, United States
*Corresponding author. Department of Computer Science, University of Montana, 32 Campus Dr, Missoula, MT 59812, United States. E-mail: daniel.olson@umontana.edu.

Associate Editor: Anna-Sophie Fiston-Lavier

Abstract

In the age of long read sequencing, genomics researchers now have access to accurate repetitive DNA sequence (including satellites) that, due to the limitations of short read-sequencing, could previously be observed only as unmappable fragments. Tools that annotate repetitive sequence are now more important than ever, so that we can better understand newly uncovered repetitive sequences, and also so that we can mitigate errors in bioinformatic software caused by those repetitive sequences. To that end, we introduce the 1.0 release of our tool for identifying and annotating locally repetitive sequence, **U**LTRA **L**ocates **T**andemly **R**epetitive **A**reas (*ULTRA*). *ULTRA* is fast enough to use as part of an efficient annotation pipeline, produces state-of-the-art reliable coverage of repetitive regions containing many mutations, and provides interpretable statistics and labels for repetitive regions.

Availability and implementation: ULTRA is released under an open source license, and is available for download at https://github.com/TravisWheelerLab/ULTRA.

1 Introduction

1.1 Background

1.1.1 Tandem repeats

Biological sequences are complex and full of subtle patterns; however, one of the most common patterns is not subtle at all: *tandem repeats*. Tandem repeats are a class of biological sequences that, for various reasons (Li *et al.* 2002, Iyer *et al.* 2015, Zhang *et al.* 2020, Zattera and Bruschi 2022), consist of multiple adjacent copies of some core repetitive unit. "AACAACAACAACAACAACA ACAAC" is an example of a short tandem repeat composed of an "AAC" unit repeated 8 times. Tandem repeats are easy to identify when their repetitive pattern is well conserved, but as a repetitive region ages and accrues mutations, the clarity of its repetitive pattern decays. For instance, "AACAACAATATCAATAACAACA ACAGCAAC" (found by *ULTRA* at location 61461 in T2T-CHM13v2.0 chromosome 1) was likely a once pristine "AAC" repeat, but after only a few substitutions, the repetitive pattern has become less obvious. Insertions and deletions further obfuscate repetitive patterns, and old tandem repeats that have experienced many mutations are frequently difficult or impossible to annotate with high confidence.

An assortment of repeat expansion mechanisms is responsible for the various classes of tandem repeats. *Short tandem repeats* (Jeffreys *et al.* 1985) are primarily caused by replication slippage (Levinson and Gutman 1987, Zhang *et al.* 2020) and have small repetitive units from 1 to 6 bp, which can repeat to span up to hundreds of bp (Fan and Chu 2007, Gymrek 2017). *Minisatellites* (Wyman and White 1980) are a larger sort of tandem repeat, having units between 6–60 bp and occupying regions as large as 20 kb (Bennett 2000). *Satellite repeats* (Kit 1961) are found in heterochromatin and can be millions of base pairs in length, having repetitive periods that range from a few bases to a few thousand bases (Garrido-Ramos 2017). *Higher-order repeats* (Willard and Waye 1987) are a subclass of satellite repeats that have repetitive units that are themselves composed complex and shifting patterns of satellite repeats (Garrido-Ramos 2017).

Tandem repeats of all sorts have long been studied for reasons of scientific intrigue and also due to their effects on human health. As a brief (and incomplete) survey: they contribute to protein and RNA function (Richards *et al.* 1993, Kajava 2012, Trigiante *et al.* 2021), they are involved in gene regulation (Nakamura *et al.* 1998, Gemayel *et al.* 2010), they influence the evolution and maintenance of centromeres and telomeres (Plohl *et al.* 2008, Melters *et al.* 2013), and they play a role in genetic diseases (Richards *et al.* 1993, Sutherland and Richards 1995, Tang *et al.* 2017, Hannan 2018).

1.1.2 Bioinformatic challenges caused by tandem repeats

Historically, locally repetitive regions of DNA caused problems with assembly (Staden 1980, Nagarajan and Pop 2013) because they are often longer than the sequence reads produced by early generations of sequencers (Pop *et al.* 2002). This challenge has been largely resolved thanks to advances in long-read sequencing (McCarthy 2010, Jain *et al.* 2016, Wenger *et al.* 2019), which produces reads that are long and accurate enough to enable assembly of centromeric satellite DNA, as exemplified by the first "complete" human reference genome, T2T-CHM13 (Altemose *et al.* 2022, Nurk *et al.* 2022).

Received: June 4, 2024; **Revised:** September 19, 2024; **Editorial Decision:** October 2, 2024; **Accepted:** October 7, 2024

Despite these gains, tandem repeats still pose substantial problems for bioinformatic analysis. Consider homology search with tools like *BLAST* (Altschul *et al.* 1990) and *HMMER* (Eddy 2011); these tools provide a means to estimate the probability that two sequences are evolutionarily related to one another (i.e. homologous). Briefly, these tools work by measuring the similarity between two sequences and then estimating how often that level of similarity is expected to occur by chance (Pagni and Jongeneel 2001). Tandem repeats occur far more frequently than expected by the simplistic random models used in homology search tools, and as a result, they are a frequent source of false positive search results. For example, consider the short tandem repeat "ATATATATATATATATATATATATATATATAT". With naive notions of random chance we would expect this exact sequence to occur once every 1.8×10^{17} nucleotides (assuming 60% AT richness), corresponding to a 6 in 10^7 chance of the sequence being found in the human genome. However, in human [T2T-CHM13v2 (Nurk *et al.* 2022)] chromosome 1, there are 624 nonoverlapping occurrences of that exact tandem repeat (as found with "`grep -oi`"), and 3998 nonoverlapping occurrences of the tandem tandem repeat with 5 or fewer substitutions (found using "`ugrep -oi -Z ~5`" on T2T-CHM13v2 Chr 1 with newlines removed). The majority of these "AT" repeats are not similar to one another because of homology, but because of independent replication slippage events that happen to produce similar short tandem repeats. Regardless of true homology, when a query sequence that contains the "AT" repeat from above is searched against human chromosome 1, that search will yield false positive hits for each nonhomologous "AT" repeat. The problem is not limited to sequences that contain perfect or near perfect tandem repeats—even highly decayed tandem repeats can and do induce high scoring false positive search results (Wheeler *et al.* 2013).

1.1.3 Reducing repeat-caused error

The most common way to alleviate bioinformatic error caused by tandem repeats is through masking (Frith 2011a; Kiełbasa *et al.* 2011). A strategy called *hard-masking* uses a tandem repeat annotation tool to find and then mask (hide) tandem repeats from all downstream analysis by replacing the repetitive letters with the ambiguous letter N (for DNA) or X (for protein).

An alternative approach, called *soft-masking*, aims to improve sensitivity relative to hard-masking. Under soft-masking, repetitive sequence is not hidden completely but is instead marked as repetitive, typically by making repetitive sequence lower-case in the representative sequence file. Some homology search tools ignore soft-masked regions during the phase of identifying alignment seeds, then allow repetitive sequence to be used while creating a final sequence alignment (Frith 2011a).

The problem with both hard-masking and soft-masking is that one must choose some threshold of repetitiveness (i.e. repetitive annotation score) to decide what should be masked and what should not be masked. Low thresholds mask too much and result in reduced annotation sensitivity (Frith *et al.* 2010). High thresholds mask too little, allowing low-scoring decayed repeats to cause false positive hits (Wheeler *et al.* 2013). One potential strategy for reducing errors caused by tandem repeats is to not mask at all, but to instead better incorporate models of repetitiveness directly into the annotation process (Nánási *et al.* 2014). Such an approach could proceed by either directly including repetitiveness in the process of identifying and scoring of alignment, or by competing repeat annotations against alignment-based annotations (Carey *et al.* 2021). Both of these approaches demand probabilistic models that produce interpretable scoring statistics.

1.2 Repeat annotation tools

There is a plethora of tools used to find tandem repeats (Benson 1999, Kurtz and Schleiermacher 1999, Sharma *et al.* 2004, Price *et al.* 2005, Jorda and Kajava 2009, Frith 2011b, Smit *et al.* 2013–2015, Ruiz-Ruano *et al.* 2016, Beier *et al.* 2017) each tool with a unique set of strengths and weaknesses. Throughout this paper we will compare our tool, *ULTRA*, against two tools in particular: *TRF* (Benson 1999) and *tantan* (Frith 2011b).

The venerable *TRF* is the most widely used *de novo* repeat-finding tool. *TRF* first searches for well conserved tandem repeat candidates, and then proceeds with a self-alignment algorithm that further extends the repeat annotation while also producing informative annotation labels describing the repetitive pattern and all mutations that occur throughout each observed tandem repeat.

tantan is based on a hidden Markov model (HMM) for repetitive sequence, and employs the Forward-Backward algorithm (Stratonovich 1965) to assign a numeric value to each letter in the target sequence representing the probability of the letter being a part of a tandem repeat. *tantan* is blazingly fast and can be tuned to provide greater sensitivity and specificity compared to *TRF*. Due to its design focus on the repetitive nature of individual letters, it does not aim to assign scores or labels to repetitive regions, and thus produces less descriptive repeat annotations than does *TRF*.

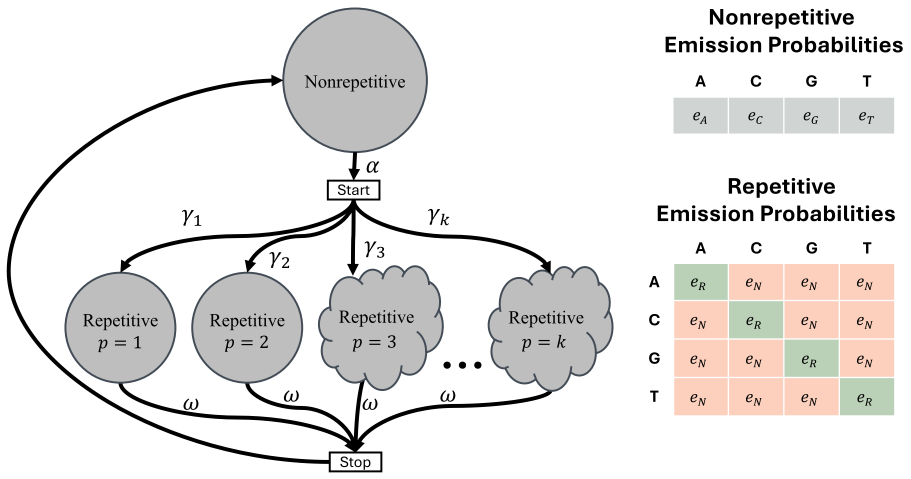
In an earlier conference paper, we described a prototype implementation of our HMM-based repeat annotation tool, **U**LTRA **L**ocates **T**andemly **R**epetitive **A**reas (*ULTRA*) (Olson and Wheeler 2018). Our goal in developing *ULTRA* is that it will improve sensitivity and specificity relative to other tools (particularly in the context of high levels of mutation), provide statistically meaningful annotation scores, create consistent and interpretable characterization of repetitive patterns, be easily re-parameterized for specific genomes, and provide a stable and easy user experience. Here, we introduce the user-ready 1.0 release of *ULTRA* and demonstrate its repeat labeling and scoring output. We also compare its performance to *TRF* and *tantan*, and show that it produces high genome coverage with low false labeling rate.

2 Methods

2.1 ULTRA's hidden Markov model

2.1.1 Repetitive and nonrepetitive states

ULTRA models tandem repeats using a hidden Markov model (HMM—for an introduction to HMMs, see (Eddy 2004, Stamp 2004)). *ULTRA*'s HMM (Fig. 1) uses a single state to represent nonrepetitive sequence, and a collection of repetitive states that each model different repetitive periodicities. The nonrepetitive state is blind to context and its emission distribution approximately represents how frequently letters are expected to occur in nonrepetitive background sequence. In contrast, each repetitive state is *context-sensitive* (Yoon and Vaidynathan 2004), meaning that the repetitive emission distribution depends on prior observations.



Nonrepetitive Emission Probabilities

A	C	G	T
e_A	e_C	e_G	e_T

Repetitive Emission Probabilities

	A	C	G	T
A	e_R	e_N	e_N	e_N
C	e_N	e_R	e_N	e_N
G	e_N	e_N	e_R	e_N
T	e_N	e_N	e_N	e_R

Figure 1. *ULTRA*'s HMM. The cloud shaped nodes represent a collection of states modeling both tandem repeats and also insertion/deletion events (see Section 2.2.1). Self-transition edges have not been drawn but do exist for the nonrepetitive and repetitive states. Similar to *tantan*, *ULTRA* models large period repeats as being less common than small period repeats through a decay parameter, λ. For a model allowing maximum period k, the probability of transitioning from the *start* state to a period p repetitive state is $\gamma_p = \lambda^p \div \sum_{i=1}^{k}(\lambda^i)$. All labeled parameters can be adjusted (see the user-guide at https://github.com/TravisWheelerLab/ULTRA).

A period p repetitive state at position t has a greater chance of repeating the letter that was observed at position $t-p$ than emitting a mismatching (nonrepetitive) letter. Importantly, the repetitive emission distribution only depends on the individual letter at position $t-p$ and is otherwise unaware of context.

2.1.2 Insertion and deletion states

Substitution mutations are modeled by the emission distribution of repetitive states. An individual substitution will cause a slight decrease in annotation score, but because it does not affect the overall repetitive pattern, it will not dramatically impact the overall repeat annotation. On the other hand, insertions and deletions (indels) cause a temporary offset in repetitive pattern and can result in a large reduction in repeat annotation score that may be significant enough to grossly change the overall repeat annotation. Consider the following example:

Sequence:	...ATACCGG	AcgTACCGG	ATACCGG...
Naive Look-back:	...0.7777777	777777777	7777777...
Naive Matches:	...+++++++	+--------	-++++++...
Correct Look-back:	...0.7777777	700999999	9777777...
Correct Matches:	...+++++++	+--++++++	+++++++...

The inserted letters "cg" are effectively nonrepetitive (i.e. period 0) because they are independent of the repetitive pattern. The insertion also causes a temporary offset in repetitive pattern, and if the indel is not accounted for, then the period p repetitive state will accumulate as many as p undeserved mismatches.

ULTRA's HMM models indel events with period-specific indel states that account not only for inserted and deleted letters, but also for the temporary offset in repetitive pattern caused by indels (see Fig. 2). *ULTRA* uses three types of states to model indels. I-states represent insertion events and are modeled as *emitting* a letter with disregard for the current repeat pattern. D-states are silent states that are modeled as *omitting* a letter in the repeat pattern. *ULTRA* uses J-states to model the offset that occurs for the p letters that follow an indel. More specifically, each I-state and D-state connects to a unique chain of p J-states that lead from the I-state or D-state back to the original period p repetitive state. During repeat annotation, an entire J-state chain can be updated efficiently without computing or storing values for individual J-states; this optimization is akin to updating a moving average based on which values have changed instead of simply recomputing the entire average. For a more complete description of this optimization see our earlier conference extended abstract (Olson and Wheeler 2018). (Note that *ULTRA*'s HMM is similar to the one used in *tantan*, supplemented by these additional indel states to explicitly model the effects of insertions and deletions. *tantan* optionally models indels, but uses complete shifts in repetitive period without requiring return to the original repetitive period.)

Modeling all possible patterns of indel events while keeping track of the temporary offset in repetitive pattern caused by those events would result in an intractably large HMM. Instead of modeling all possible patterns, *ULTRA* only models up to i consecutive insertions and up to d consecutive deletions where, by default, $i = d = 10$. Anecdotally, we find that more complex indel patterns can frequently be well approximated by *ULTRA*'s basic model of consecutive indels, and empirically we find that modeling consecutive indels provides improved annotation performance relative to modeling no indels at all.

2.2 Viterbi annotation

2.2.1 Basic viterbi

To annotate repeats in a length n string S, *ULTRA* finds regions of S that are well represented by repetitive states and then marks those regions as being repetitive. The problem of finding the sequence of HMM states with the greatest probability of producing an observed string is known as the *decoding problem*, and *ULTRA* solves that problem using the *Viterbi algorithm* (Viterbi 1967). Let Σ be the model alphabet (e.g. in DNA, $\Sigma = \{A, C, G, T\}$), M be the collection of m *ULTRA* HMM states (including nonrepetitive, repetitive, I, D, and J states), $T \in R^{m \times m}$ be the transition probabilities, and $E \in R^{|\Sigma|^2 \times m}$ be the emission probabilities; we will use the shorthand $E_{t,v}$ to refer to the probability of state v emitting

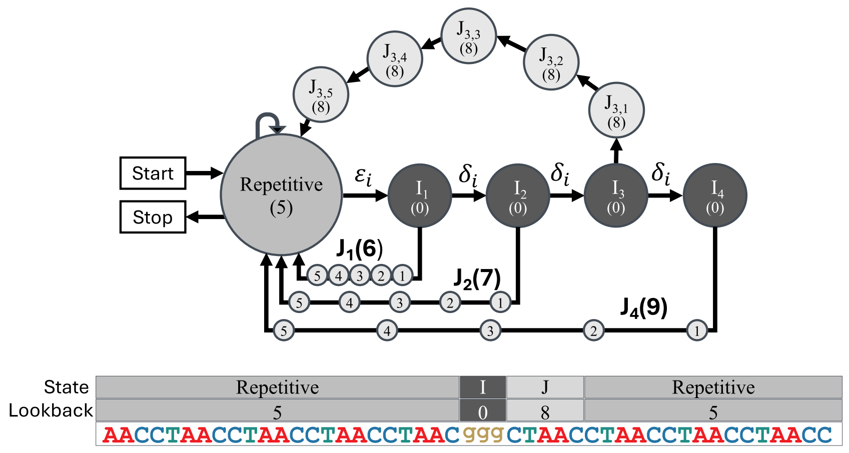


Figure 2. Top: The collection of states used to model insertions that occur within a $p = 5$ tandem repeat (a similar collection of states is used to model deletions). Each state's look-back is shown within parentheses. Bottom: A $p = 5$ tandem repeat that contains a length 3 insertion. The letters from the insertion are explained by a path through 3 I-states with look-back = (0), followed by a chain of J-states with look-back = 3 + 5 = (8).

the letter $S_t \in \Sigma$ at position t. Viterbi works by creating a matrix, $P \in R^{n\times m}$, i.e. filled using the following recurrence:

$$P_{t,v} \leftarrow \begin{cases} 0 & t=0, v\neq 0 \\ 1 & t=0, v=0 \\ \max_{u\in M} P_{t-1,u} T_{u,v} E_{t,v} & 0<t\leq n \end{cases} \quad (1)$$

Let v^* be state with the greatest value in P_n, i.e. $v^* = \text{argmax}_{u\in M} P_{n,u}$. The probability inside P_{n,v^*} will be the probability of a most likely path through M, i.e. the *Viterbi path*. The Viterbi path can then be recovered by re-tracing the transitions which led to v^* at index n [see (Stamp 2004) for a more complete description of Viterbi].

2.2.2 ULTRA viterbi

ULTRA's Viterbi implementation replaces emission probabilities with the ratio of model emission probability relative to the background frequency of letters, E_0. We refer to *ULTRA*'s Viterbi matrix as P'.

$$P'_{t,v} \leftarrow \begin{cases} 0 & t=0, v\neq 0 \\ 1 & t=0, v=0 \\ \max_{u\in M} P'_{t-1,u} T_{u,v} \dfrac{E_{t,v}}{E_{t,0}} & 0<t\leq n \end{cases} \quad (2)$$

The effects of using relative emission likelihood are greatest in highly biased sequence composition, and enable *ULTRA* to better differentiate repeating letters that occur by random chance (because the letters are very common) from repeating letters that are part of a tandem repeat. Importantly, *ULTRA*'s usage of relative emission likelihood inside of Viterbi produces very different results from using basic Viterbi followed by a composition-bias adjustment; in testing we observed that using relative likelihood of emissions greatly improved *ULTRA*'s overall accuracy.

One downside to using relative emission likelihood is that it is more difficult to predict how adjustments to the background emission rate parameters will affect repeat annotation. In future updates to *ULTRA* we wish to improve the interpretability of *ULTRA*'s background probability rate, but until then we suggest that users take advantage of the `–tune` subprogram (described in Section Tuning), which automatically optimizes *ULTRA*'s emission probabilities for a given input sequence.

2.2.3 Using the viterbi path to annotate repeats

After the Viterbi path, $V = \{v_0, v_1, \ldots, v_n\}$, has been calculated, it is then used to annotate tandem repeats. A substring with range $(t_{\text{start}}, t_{\text{end}})$ will be annotated as repetitive when all of the positions in the range are identified by the Viterbi path as having been emitted by a repetitive state (or its associated indel states). In other words, a region is labeled as repetitive when $\forall t \in \{t_{\text{start}}, \ldots, t_{\text{end}}\}, v_t \neq 0$. After a p period repeat has been found in the range $(t_{\text{start}}, t_{\text{end}})$, it is then scored using the following equation:

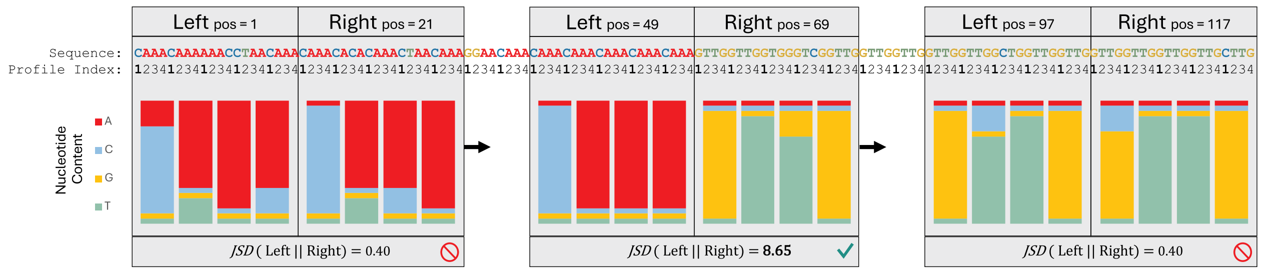
$$\begin{aligned} \text{score} &= \sum_{\mathcal{P}t=t_{\text{start}}}^{t_{\text{end}}} \left(\log(T_{v_{t-1},v_t}) + \log\left(\frac{E_v(t)}{E_0(t)}\right)\right) \\ &= \log(P'_p(t_{\text{end}})) - \log(P'_p(t_{\text{start}})) \end{aligned} \quad (3)$$

Because repetitive states in *ULTRA* look backwards but not forwards, the first p letters of a repeat will not be part of the Viterbi path. *ULTRA* remedies this problem by reporting the starting location for repeats as $t_{\text{start}} - p$ instead of t_{start}.

2.3 Repeat splitting

"ACACACACACACGCGCGCGCGCGCGC" is a tandem repeat that changes from an "AC" pattern to a "GC" pattern; because repetitive states in *ULTRA*'s repeat model only compare the letter at position t to the letter at position $(t-p)$, *ULTRA* only observes a single mismatch when the pattern changes and is therefore unaware of the change in pattern. To annotate pattern changes such as this, *ULTRA* performs a post processing *repeat splitting* step. We refer to locations where a change in pattern occurs as *repeat splits*, and we refer to the segments of repetitive sequence isolated by repeat splits as *subrepeats*.

ULTRA's repeat splitting process (see Fig. 3) begins by assigning *profile-indices* to letters in the sequence. The profile-indices are used to keep track of how letters in the repetitive sequence relate to the repetitive pattern. For example, the sequence "CATCATCAgTCATCAT" is assigned the indices "**1** 2 3 **1** 2 3 **1** 2 * 3 **1** 2 3 **1** 2 3". *ULTRA* then slides two adjacent windows along the tandem repeat and uses the sequence and


Left pos = 1
Right pos = 21
Left pos = 49
Right pos = 69
Left pos = 97
Right pos = 117
Sequence: CAAACAAAAAACCTAACAAACAAACACACAAACTAACAAAGGAACAAACAAACAAACAAACAAACAAAGTTGGTTGGTGGGTCGGTTGGTTGGTTGGTTGGTTGGCTGGTTGGTTGGTTGGTTGGTTGGTTGCTTG
Profile Index: 1234123412341234123412341234123412341234123412341234123412341234123412341234123412341234123412341234123412341234123412341234123412341234
Nucleotide Content
A
C
G
T
JSD (Left || Right) = 0.40
JSD (Left || Right) = 8.65
JSD (Left || Right) = 0.40

```
Repeat: "AAAC GGTT", Start: 1, End: 136, Score:108.0
Subrepeat: "AAAC",     Start:1,  End:69,   Score: 54.4    Subrepeat: "GGTT",    Start:69,  End:136,   Score: 53.6
```

Figure 3. A period 4 repeat with two subrepeats ("AAAC" and "GGTT"), each containing multiple substitutions. To find the change in pattern, *ULTRA* slides two adjacent windows along the sequence and creates profiles representing the repetitive content within the windows. The repetitive profiles of two adjacent windows are compared against each other using JSD. This figure shows the local window profiles (with profile frequencies displayed as bar charts) and the corresponding JSD for three different positions. The first pair of window profiles contains similar repetitive content resulting in a small JSD; the last profile pair also yields a small JSD. The middle windows contain different repetitive content, resulting in a large JSD that passes *ULTRA*'s splitting threshold. Both the repetitive region as a whole and also the repetitive region's subrepeats are included in *ULTRA*'s final annotation.

profile indices to build window-specific repeat-profiles, L and R. Each window must be at least (by default) 15 letters long and at least (by default) 5 repeat units. The window profiles are stored as matrices; a period p repeat with an alphabet Σ will have a corresponding profile matrix of size $p \times |\Sigma|$. Each $L_{i,c}$ and $R_{i,c}$ cell holds the frequency with which letter c is observed at profile-index i in the window. The repetitive content of L is then compared against the repetitive content of R by calculating the Jensen-Shannon divergence (Jeffreys 1946, Lin 1991):

$$\mathbf{JSD}(L||R) = -\frac{1}{2}\sum_{i=1}^{p}\sum_{c\in M} L_{i,c}\log\left(\frac{R_{i,c}}{L_{i,c}}\right) + R_{i,c}\log\left(\frac{L_{i,c}}{R_{i,c}}\right) \tag{4}$$

ULTRA slides the two repeat profile windows across the entire repetitive sequence and iteratively calculates the JSD at each position in the repetitive sequence. A small JSD implies that both windows contain similar repetitive content and can be well represented as a single repetitive region. Alternatively, a large JSD means that the left and right windows contain dissimilar repetitive content. ULTRA identifies local peaks in JSD, and when the value of a peak exceeds a threshold (3.5 by default), the corresponding position is marked as a repeat split. After splits have been marked, the repetitive patterns are calculated for each subrepeat.

In cases where an undetected indel has occurred, the profile indices will be incorrectly assigned resulting in a large JSD even when the repetitive pattern has not actually changed. *ULTRA* accounts for this possibility by comparing neighboring subrepeats to ensure the subrepeats have different repetitive patterns. When two neighboring subrepeats have the same (possibly rotated) repetitive pattern (such as "GTTG" and "GGTT"), the repeat split is removed and the two subrepeats are merged back into a single subrepeat. The final annotations output by *ULTRA* describe the overall repetitive region as well as the subrepeats that make up the region. Repeats and subrepeats are named with an alphabetically sorted version of their repetitive pattern (e.g. the pattern "GGTT" from Fig. 3 is alphabetically smaller than rotations "GTTG," "TTGG," and "TGGT").

2.4 Calculating *P*-values

It is possible to base repeat annotations directly on scores, by establishing some score threshold τ and identifying a candidate region as repetitive if its *ULTRA* score exceeds τ. But sequence annotation is more robust when scores can be interpreted in a likelihood framework; this is common for alignment-based annotation, where each alignment with score s is assigned a P-value (and multiple-test-adjusted E-value) that corresponds to the probability of observing a score $\geq s$ in an alignment of two unrelated sequences. *ULTRA* similarly computes a P-value for each annotated region by learning the distribution of scores observed in random (nonrepetitive) sequence.

ULTRA was used (with default settings) to annotate repetitive regions in 1 GB of randomly generated 50% AT-rich sequence. We then fit exponential distribution parameters (location $= \mu$, and scale $= \sigma$) to the distribution of annotation scores observed in the 1 GB random sequence. Letting ω be the fraction of letters annotated as repetitive, and s be the *ULTRA* annotation score of a labeled repetitive region, the P-value of that region can be calculated as $\mathtt{P-value(score}s\mathtt{)} = \omega\exp((\mu - s)/\sigma)$.

The `–pval` flag will instruct *ULTRA* to convert annotation scores to P-values. We note that changes in *ULTRA's* model parameters or changes in sequence composition can greatly effect the location, scale, and repeat frequency parameters that should be used to produce P-values. In a future release, we plan to automatically extract location, scale, and repeat frequency during tuning (see Section 2.6). Until then, users may modify P-value parameters with the `–ploc`, `–pscale`, and `–pfreq` options.

2.5 Windowed viterbi

Applying Viterbi to a length n sequence using an HMM with m states requires storing values in an $n \times m$ matrix. The size of that matrix can be a limiting factor when analyzing large sequences with large models. To reduce the amount of memory required, *ULTRA* splits an input sequence into overlapping windows and then applies Viterbi to each window independently. Dividing the target sequence into windows has the added benefit of enabling parallel computing of repeats across distinct windows. Using overlapping windows introduces the possibility of tandem repeats that occur in the overlapping region of windows being annotated multiple times. To avoid redundant repeat annotation, *ULTRA* will search for and then merge overlapping repeats when they are equal in repetitive period.

ULTRA automatically adjusts the size of sequence windows and the amount of overlap between windows based on the number of states *ULTRA*'s HMM is using; when using a large collection of states the window size is decreased (in order to reduce memory usage) and when using a small collection of states the window size is increased (to improve runtime). Window size and overlap size can also be manually adjusted with the `–winsize` and `–overlap` options. Users can see how much memory *ULTRA* is expected to consume by using the `–mem` flag.

2.6 Tuning

There is tremendous diversity in tandem repeat patterns between different organisms. To achieve the highest annotation quality, *ULTRA*'s parameters need to be tuned according to the genome being annotated. In *ULTRA*, parameter tuning is performed automatically when used with the `–tune` flag. Tuning works by running *ULTRA* many times on the input sequence, each time using a different set of candidate parameters. *ULTRA* then selects the parameter set that provides the greatest annotation coverage while remaining under an estimated false discovery rate threshold (10% by default).

To estimate the false discovery rate of a particular parameter set, *ULTRA*'s tuning routine measures *ULTRA*'s annotation coverage on a locally shuffled version of the input sequence. *ULTRA*'s local shuffling is done inside the sequence windows used by windowed Viterbi (see Section Windowed Viterbi). Local shuffling retains local variability in sequence composition, as is observed in isochores (Bernardi *et al.* 1985, Eyre-Walker and Hurst 2001, Costantini and Musto 2017), but removes all biologically caused tandem repeats. By chance, the shuffled sequence will still contain some repetitive content, but tandem repeats that occur in shuffled sequence are expected to be rare and low-scoring relative to the original (biological) input sequence. After *ULTRA* has used candidate parameters to annotate both the input sequence and the shuffled input sequence, it then estimates the false discovery rate as repetitive coverage in the shuffled sequence divided by the repetitive coverage in the original input sequence.

ULTRA allows users to change what parameter configurations will be tested during tuning. `–tune` explores 18 different parameter configurations, `–tune_medium` explores 40, and `–tune_large` explores 252. Users can also provide their own candidate parameter configurations using the `–tune_file` option. By default, *ULTRA* reduces the runtime of tuning by disabling indel states. After indel-free identification of good parameters, *ULTRA* is re-run with indels enabled. This approach can sometimes lead to selection of parameters that produce higher-than-expected false discovery rate once the indel model is enabled in the final run; to bypass this risk, indel states can be enabled during tuning with the `–tune_indel` flag.

3 Results

3.1 Coverage

We quantify the accuracy of *ULTRA*, *TRF*, and *tantan* by measuring each tool's coverage and estimated false coverage for the genomes listed in Table 1. In order to estimate false coverage, we ran each tool on a window-shuffled version of each genome. Specifically, we used the *esl-shuffle* tool from *HMMER*'s (Eddy 2011) easel library, applying the "`-w 20000`" flag to shuffle 20 kb chunks so that regional GC compositional variability would be retained while also removing (through shuffling) biologically caused tandem repeats. We use annotation coverage of the shuffled genomes to estimate the false coverage of the unshuffled genomes.

Table 1. Genomes used for coverage experiments.

Organism	Strain/version	GC %
Mycobacterium tuberculosis	H37Rv (AL123456.2) (Cole *et al.* 1998)	65.6%
Neurospora crassa	NC12 (Galagan *et al.* 2003)	48.2%
Zea Maize	B73-REFERENCE-NAM-5.0 (Hufford *et al.* 2021)	46.7%
Homo sapien	T2T-CHM13v2.0 (Nurk *et al.* 2022)	40.8%
Plasmodium falciparum	ASM276v2 (Gardner *et al.* 2002)	19.3%

Figure 4 shows each tool's annotation coverage for each genome using default settings (for *TRF* we used "`2 7 7 80 10 30` < max repeat period > `-l 12`", as suggested in the TRF user guide). We also report results achieved with settings optimized for each organism; these settings were identified by performing a parameter grid search (see below). Finally, we show coverage for *ULTRA* using the "`–tune`" subprogram, which causes *ULTRA* to perform an automated grid search (described in Section Tuning). The exact coverage values and selected (optimized) parameters can be found in Supplementary Tables S1–S8.

For the optimized coverage experiments, we created a grid of parameters for each tool (see Table 2), and then for each genome, we selected the parameters that produced the highest coverage with an estimated false discovery rate of 10% or less.

By default, *tantan* will annotate repetitive sequence even if there are fewer than two full repeat units, causing *tantan* to sometimes use a large repeat period to annotate noncontiguous repetitive content (i.e. content that is not *tandemly* repetitive). To make *tantan*'s results more consistent with *ULTRA* and *TRF*, we used "*tantan* `-f4`" for all coverage experiments, which (among other things) causes *tantan* to filter out repeat annotations that are less than two repetitive units in length. We also measured *tantan*'s performance without `-f4` and compare it against *ULTRA* with comparable settings (`–min_unit` 0) and found that "*ULTRA* `–min_unit` 0" has greater coverage than *tantan* without the `-f4` flag, with similar relative results (see Supplementary Figs S1–S4).

For both default settings and optimized settings, *ULTRA* produces greater coverage than *TRF* and *tantan*. For most genomes, *ULTRA* also had the smallest FDR with the notable exception being *Plasmodium falciparum*, caused by the genome's highly biased 80.7% AT-rich composition, which is far from *ULTRA*'s default composition expectation. For many genomes, using *ULTRA* with `–tune` produced a larger FDR compared to *ULTRA* (default), especially in the period 500 coverage experiments. The increased FDR while tuning is primarily caused by `–tune` disabling indel states during tuning; for best results we suggest users use `–tune_indel`.

3.2 Repeat score distribution

When using repeat masking strategies, one must choose a score threshold that will determine which regions will and will not be masked. Understanding the distribution of annotation scores in random sequence allows researchers to make statistically informed decisions about which score threshold

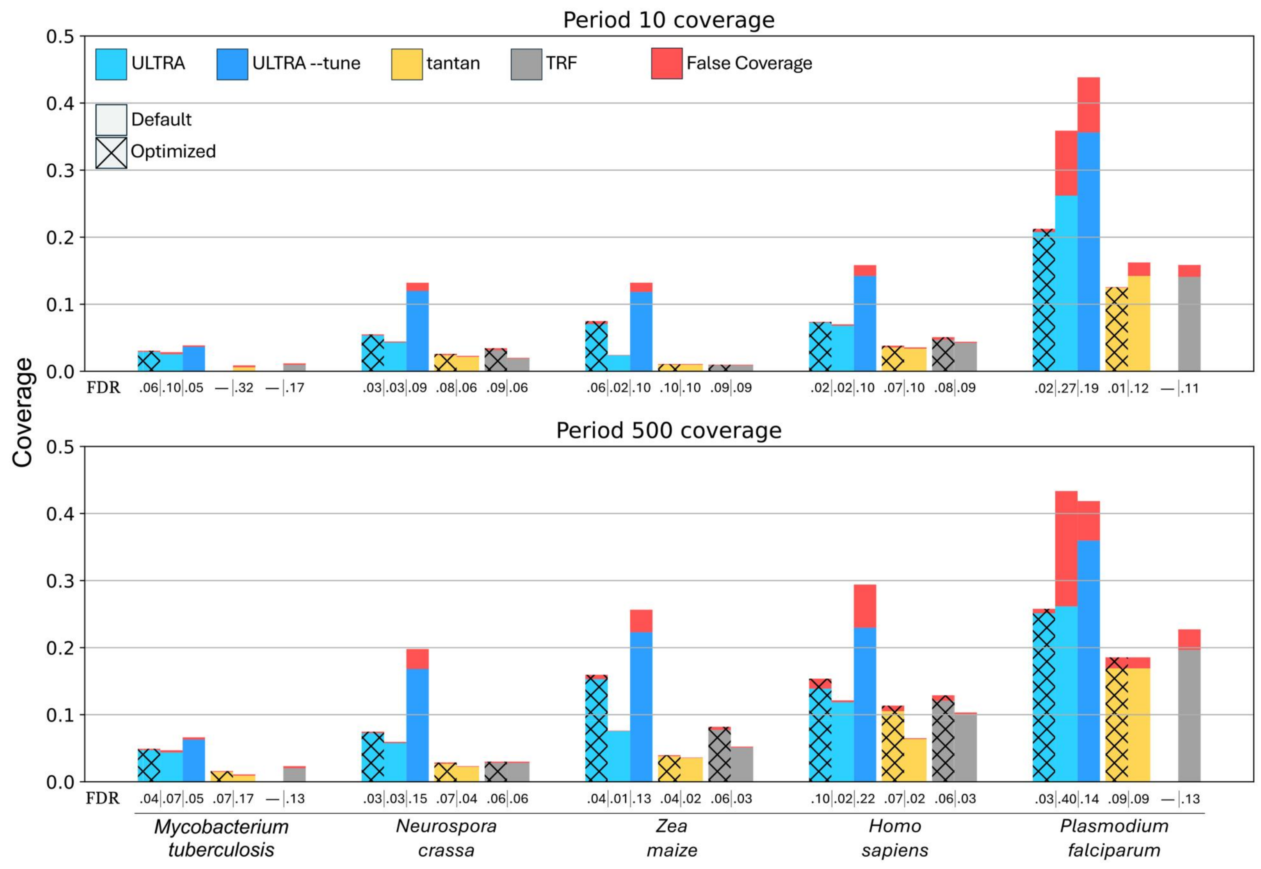


Figure 4. Coverage and estimated false coverage for *ULTRA*, *tantan*, and *TRF*. The top chart show coverage when using a maximum repeat period of 10 and the bottom chart show coverage when using a maximum repeat period of 500. Plain bars indicate default parameters and textured bars indicate grid-search optimized parameters. We also include results using *ULTRA* `–tune` (with default settings). The estimated false discovery rate (FDR) is displayed below each bar. Note that in some case, there is no parameter choice that achieves <10% FDR; in these cases, no bar is presented, and the FDR value is listed as —.

Table 2. Parameter grids for optimized coverage experiments.[a]

Tool	Parameter	Values
ULTRA	`--at` AT richness	0.3, 0.4, 0.5, 0.6, 0.7
	`–m` Repeat match probability (Pr [$S_t = S_{t-p}$])	0.6, 0.7, 0.9
tantan	`–r` Repeat start probability	0.005, 0.1
	`–s` Repeat annotation threshold	0.50, 0.85
	`-m` Substitution matrix	(default) 50% GC richness substitution matrix, 78% AT richness substitution matrix, 78% GC richness substitution matrix.
TRF	"2 7 7 80 10 30 < max repeat period > -l 12"	
	"2 5 5 80 10 30 < max repeat period > -l 12".	

[a] *ULTRA*'s grid is composed of 15 total parameter combinations, *tantan*'s grid has 12 parameter combinations, and *TRF* grid has 2 different configurations, one from the TRF user guide, and one from Frith (2011b).

they should use, and may enable more sophisticated repeat-error mitigation than mere masking [such as competing annotations based on score density or associated statistical values (Carey *et al.* 2021)]. Additionally, if a tool is incapable of producing well behaving score distributions when annotating random sequence, then it is likely that the tool will provide inconsistent scores to biological tandem repeats, resulting in lower quality repeat masking.

Figure 5 shows the score distributions of *ULTRA*, *tantan*, and *TRF* when run on 10 GB of randomly generated 60% AT-rich sequence. We ran all three tools using default settings and a maximum detectable repeat period of 100. Its important to note that while *ULTRA* and *TRF* produce scores for each tandem repeat annotation, *tantan* instead produces probabilities (Forward-Backward derived posterior marginals) for individual letters.

ULTRA's score distribution decays smoothly and exponentially, while *TRF*'s scores decay chaotically. *tantan*'s probability distribution cannot be compared directly to the scores produced by *ULTRA* or *TRF* because they represent letter-specific probabilities of being part of some tandem repeat instead of tandem repeat annotation scores, but we note that *tantan*'s posterior marginal distribution decays very smoothly.

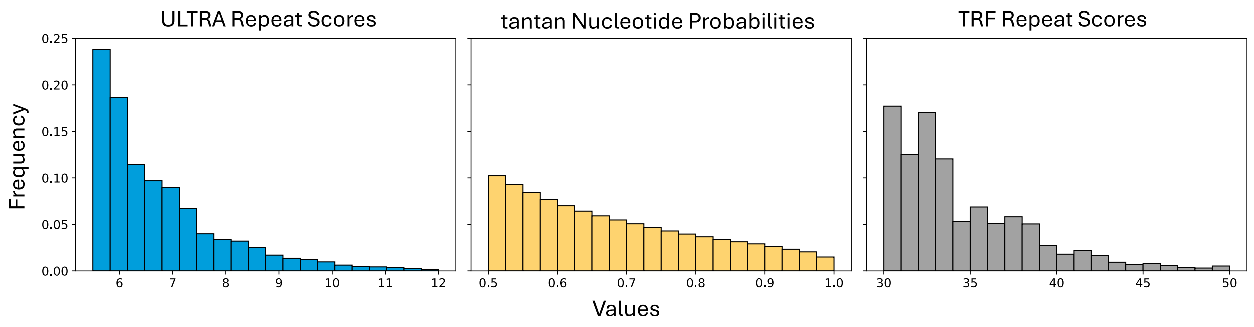


Figure 5. Annotation score distributions. Using each tool to label 10 GB of 60% AT-rich random sequence, the left and right plots show per-repeat score distributions for *ULTRA* and *TRF*, respectively. The middle plot shows the distribution of emphtantan per-letter probabilities of being part of a repetitive region. Horizontal axis corresponds to score/probability values and the vertical axis corresponds to value frequency. The exponential decay of ULTRA enables reliable *P*-value estimates.

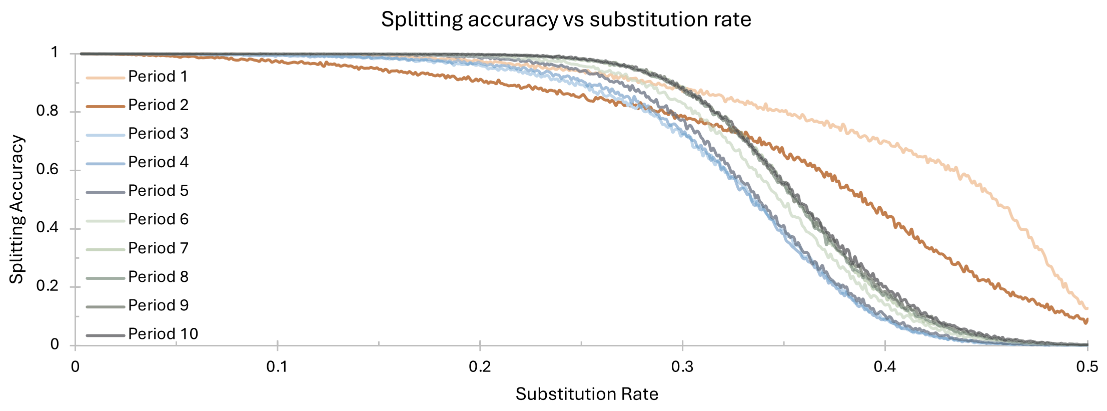


Figure 6. Repeat splitting accuracy versus sequence substitution rate.

3.3 Repeat splitting accuracy

Repeat splitting (described in Section 2.2.3) allows *ULTRA* to find and annotate changes in repetitive pattern. Figure 6 shows *ULTRA*'s repeat splitting accuracy for repetitive periods 1 through 10 under substitution rates between 0 and 0.5. To measure repeat splitting accuracy, we generated artificial repetitive sequences that contained two subrepeats. Sequences were made by first selecting a repeat period between 1 and 10, and then creating two repeat units (such as "AACC" and "ACGT") that shared no >50% similarity. The similarity restriction considers all rotations of the repetitive units; e.g. "ACCGG" and "GGAGG" would *not* be a valid pair of subrepepeat patterns because "ACCGG" rotated two letters to the right is "GGACC", which shares >50% similarity with "GGAGG".

We then generated two perfect tandem repeats of length 500, one for each repetitive unit. We concatenated the two perfect tandem repeats, creating one sequence of length 1000. We then mutated the sequence using a substitution rate ranging between 0 and 0.5, where each mutated letter is equally likely to mutate to any other letter. For each combination of repeat period and substitution rate, we generated 5000 sequences and then ran *ULTRA* on each sequence. We quantify *ULTRA*'s repeat splitting accuracy as the fraction of sequences in which *ULTRA* found a single change in repetitive pattern that was within 10 nucleotides of the true change in repeat pattern. Figure 6 demonstrates that *ULTRA*'s repeat splitting is generally accurate for repetitive sequence with a substitution rate of 30% or less. This suggests that it may be appropriate to report repeat units for subrepeats estimated to contain sequence identity >70%, and to otherwise report a core "ambiguous" repeat unit for the entire block.

3.4 Computational resources

Table 3 shows memory usage and runtime while processing the T2T genome (3.2 GB) with *ULTRA*, *TRF*, and *tantan*. All analysis was performed on 2.4 GHz XE2242 CPUs with 94 cores and 512 GB of RAM. *TRF* with a maximum detectable repeat period of 10 failed to process all T2T chromosomes without crashing despite efforts to adjust the "–l" option and increase *TRF*'s allocated memory. *ULTRA* is the only tool capable of multithreading, and all of *ULTRA*'s analysis was performed using 16 threads. *ULTRA*'s memory footprint can linearly be decreased by reducing the number of threads used.

4 Discussion

Here, we have introduced our open-source tool for identifying and labeling repetitive DNA: *ULTRA* (https://github.com/TravisWheelerLab/ULTRA), in hopes that it both aids the study of tandem repeats and also helps reduce the bioinformatic problems caused by tandem repeats. A prototype implementation of *ULTRA* was used during the development of T2T-CHM13 (Altemose *et al.* 2022) and the 1.0 version of *ULTRA* will soon be integrated into *RepeatMasker* (Smit *et al.* 2013–2015) (replacing *TRF*). We plan to continue supporting and improving *ULTRA* and have already strategized algorithmic tweaks and model changes that will further

Table 3. Computational time/memory used while annotating the T2T genome.[a]

Tool (period)	User time (h)	Wall time (h)	Peak memory usage (GB)
ULTRA$_{10}$ (10)	1.20	0.08	1.1
ULTRA$_5$ (10)	0.89	0.06	1.0
ULTRA$_0$ (10)	0.16	0.02	0.7
tantan (10)	0.02	0.02	1.3
TRF (10)	8.66	8.66	6.1
ULTRA$_{10}$ (500)	114.83	7.17	3.7
ULTRA$_5$ (500)	60.80	3.82	2.1
ULTRA$_0$ (500)	6.50	0.40	2.0
tantan (500)	0.56	0.56	1.3
TRF (500)	10.20	10.20	19.6

[a] Default settings were used for *ULTRA*$_{10}$, *TRF*, and *tantan*. *ULTRA*$_5$ uses 5 insertion/deletion states, and *ULTRA*$_0$ uses no insertion/deletion states. *TRF* (10) crashed before annotating all chromosomes.

increase *ULTRA*'s speed, sensitivity, and specificity. We also highlight that *ULTRA* 1.0 supports only DNA and RNA sequences; support for protein sequences will be added in a future release.

The patterns of tandem repeats are complex and far from fully understood. Higher order repeats are particularly complex, and there is great need for methods that characterize their hierarchical structure and repetitive patterns. As we continue supporting and developing *ULTRA* we hope to expand upon the descriptive annotations currently provided by *ULTRA* so that repetitive sequence can be better characterized and better understood.

Acknowledgements

We thank Robert Hubley, Arian Smit, and Tim Anderson for helpful discussions during development of software and benchmarks. We also gratefully acknowledge the computational resources and expert administration provided by the University of Montana's Griz Shared Computing Cluster (GSCC), CyVerse's External Collaborative Partnership program (provided by Tyson Swetnam), and the high performance computing (HPC) resources supported by the University of Arizona TRIF, UITS, and Research, Innovation, and Impact (RII) and maintained by the UArizona Research Technologies department.

Author contributions

Daniel R. Olson (Conceptualization [equal], Investigation [lead], Methodology [lead], Software [lead], Validation [lead], Visualization [lead], Writing—original draft [lead], Writing—review & editing [equal]) and Travis J. Wheeler (Conceptualization [equal], Data curation [supporting], Funding acquisition [lead], Methodology [supporting], Project administration [lead], Resources [lead], Supervision [lead], Visualization [supporting], Writing—original draft [supporting], Writing—review & editing [equal])

Supplementary data

Supplementary data are available at *Bioinformatics Advances* online.

Conflict of interest

None declared.

Funding

This work was supported by the National Institute of Health (NIH) National Institute of General Medical Sciences (NIGMS) [P20GM103546 and R01GM132600]. D.R.O. was additionally supported by the National Institute of Allergy and Infectious Diseases (NIAID), National Institutes of Health, Department of Health and Human Services under the Bioinformatics and Computational Biosciences Branch.

Data availability

Additional supplementary materials can be found at https://github.com/TravisWheelerLab/ULTRA-paper, including code for analysis of data.

References

Altemose N, Logsdon GA, Bzikadze AV *et al.* Complete genomic and epigenetic maps of human centromeres. *Science* 2022; **376**:eabl4178.

Altschul SF, Gish W, Miller W *et al.* Basic local alignment search tool. *J Mol Biol* 1990;**215**:403–10.

Beier S, Thiel T, Münch T *et al.* MISA-web: a web server for microsatellite prediction. *Bioinformatics* 2017;**33**:2583–5.

Bennett P. Demystified…: microsatellites. *Mol Pathol* 2000;**53**:177.

Benson G. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res* 1999;**27**:573–80.

Bernardi G, Olofsson B, Filipski J *et al.* The mosaic genome of warm-blooded vertebrates. *Science* 1985;**228**:953–8.

Carey KM, Hubley R, Lesica GT *et al.* PolyA: a tool for adjudicating competing annotations of biological sequences. bioRxiv, 2021, preprint: not peer reviewed. https://doi.org/10.1101/2021.02.13.430877.

Cole ST, Brosch R, Parkhill J *et al.* Deciphering the biology of mycobacterium tuberculosis from the complete genome sequence. *Nature* 1998;**396**:190.

Costantini M, Musto H. The isochores as a fundamental level of genome structure and organization: a general overview. *J Mol Evol* 2017;**84**:93–103.

Eddy SR. What is a hidden Markov model? *Nat Biotechnol* 2004; **22**:1315–6.

Eddy SR. Accelerated profile HMM searches. *PLoS Comput Biol* 2011; **7**:e1002195.

Eyre-Walker A, Hurst LD. The evolution of isochores. *Nat Rev Genet* 2001;**2**:549–55.

Fan H, Chu J-Y. A brief review of short tandem repeat mutation. *Genomics Proteomics Bioinf* 2007;**5**:7–14.

Frith MC. Gentle masking of low-complexity sequences improves homology search. *PLoS ONE* 2011a;**6**:e28819.

Frith MC. A new repeat-masking method enables specific detection of homologous sequences. *Nucleic Acids Res* 2011b;**39**:e23.

Frith MC, Hamada M, Horton P. Parameters for accurate genome alignment. *BMC Bioinformatics* 2010;**11**:80.

Galagan JE, Calvo SE, Borkovich KA *et al.* The genome sequence of the filamentous fungus neurospora crassa. *Nature* 2003;**422**:859–68.

Gardner MJ, Hall N, Fung E *et al.* Genome sequence of the human malaria parasite plasmodium falciparum. *Nature* 2002;**419**:498–511.

Garrido-Ramos MA. Satellite DNA: an evolving topic. *Genes (Basel)* 2017;**8**:230.

Gemayel R, Vinces MD, Legendre M *et al.* Variable tandem repeats accelerate evolution of coding and regulatory sequences. *Annu Rev Genet* 2010;**44**:445–77.

Gymrek M. A genomic view of short tandem repeats. *Curr Opin Genet Dev* 2017;**44**:9–16.

Hannan AJ. Tandem repeats mediating genetic plasticity in health and disease. *Nat Rev Genet* 2018;**19**:286–98.

Hufford MB, Seetharam AS, Woodhouse MR *et al.* De novo assembly, annotation, and comparative analysis of 26 diverse maize genomes. *Science* 2021;**373**:655–62.

Iyer RR, Pluciennik A, Napierala M *et al.* DNA triplet repeat expansion and mismatch repair. *Annu Rev Biochem* 2015;**84**:199–226.

Jain M, Olsen HE, Paten B *et al.* The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. *Genome Biol* 2016;**17**:1–11.

Jeffreys AJ, Wilson V, Thein SL. Hypervariable 'minisatellite' regions in human DNA. *Nature* 1985;**314**:67–73.

Jeffreys H. An invariant form for the prior probability in estimation problems. *Proc R Soc Lond Ser A Math Phys Sci* 1946; **186**:453–61.

Jorda J, Kajava AV. T-REKS: identification of Tandem REpeats in sequences with a K-meanS based algorithm. *Bioinformatics* 2009; **25**:2632–8.

Kajava AV. Tandem repeats in proteins: from sequence to structure. *J Struct Biol* 2012;**179**:279–88.

Kiełbasa SM, Wan R, Sato K *et al.* Adaptive seeds tame genomic sequence comparison. *Genome Res* 2011;**21**:487–93.

Kit S. Equilibrium sedimentation in density gradients of DNA preparations from animal tissues. *J Mol Biol* 1961;**3**:711–6.

Kurtz S, Schleiermacher C. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics* 1999;**15**:426–7.

Levinson G, Gutman GA. Slipped-strand mispairing: a major mechanism for DNA sequence evolution. *Mol Biol Evol* 1987;**4**:203–21.

Li Y-C, Korol AB, Fahima T *et al.* Microsatellites: genomic distribution, putative functions and mutational mechanisms: a review. *Mol Ecol* 2002;**11**:2453–65.

Lin J. Divergence measures based on the shannon entropy. *IEEE Trans Inform Theory* 1991;**37**:145–51.

McCarthy A. Third generation DNA sequencing: pacific biosciences' single molecule real time technology. *Chem Biol* 2010;**17**:675–6.

Melters DP, Bradnam KR, Young HA *et al.* Comparative analysis of tandem repeats from hundreds of species reveals unique insights into centromere evolution. *Genome Biol* 2013;**14**:R10.

Nagarajan N, Pop M. Sequence assembly demystified. *Nat Rev Genet* 2013;**14**:157–67.

Nakamura Y, Koyama K, Matsushima M. VNTR (variable number of tandem repeat) sequences as transcriptional, translational, or functional regulators. *J Hum Genet* 1998;**43**:149–52.

Nánási M, Vinař T, Brejová B. Probabilistic approaches to alignment with tandem repeats. *Algorithms Mol Biol* 2014;**9**:11.

Nurk S, Koren S, Rhie A *et al.* The complete sequence of a human genome. *Science* 2022;**376**:44–53.

Olson D, Wheeler T. ULTRA: a model based tool to detect tandem repeats. In: *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, Washington, D.C., USA*. 2018, 37–46.

Pagni M, Jongeneel CV. Making sense of score statistics for sequence alignments. *Brief Bioinform* 2001;**2**:51–67.

Plohl M, Luchetti A, Meštrović N *et al.* Satellite DNAs between selfishness and functionality: structure, genomics and evolution of tandem repeats in centromeric (hetero) chromatin. *Gene* 2008;**409**:72–82.

Pop M, Salzberg SL, Shumway M. Genome sequence assembly: algorithms and issues. *Computer* 2002;**35**:47–54.

Price AL, Jones NC, Pevzner PA. De novo identification of repeat families in large genomes. *Bioinformatics* 2005;**21**:i351–8.

Richards RI, Holman K, Yu S *et al.* Fragile X syndrome unstable element, p (CCG) n, and other simple tandem repeat sequences are binding sites for specific nuclear proteins. *Hum Mol Genet* 1993;**2**:1429–35.

Ruiz-Ruano FJ, López-León MD, Cabrero J *et al.* High-throughput analysis of the satellitome illuminates satellite DNA evolution. *Sci Rep* 2016;**6**:28333–14.

Sharma D, Issac B, Raghava GPS *et al.* Spectral repeat finder (SRF): identification of repetitive sequences using fourier transformation. *Bioinformatics* 2004;**20**:1405–12.

Smit AFA, Hubley R, Green P. RepeatMasker Open-4.0, 2013–2015. http://www.repeatmasker.org (November 2024, date last accessed).

Staden R. A mew computer method for the storage and manipulation of DNA gel reading data. *Nucleic Acids Res* 1980;**8**:3673–94.

Stamp M. *A Revealing Introduction to Hidden Markov Models*. San Jose, California: Department of Computer Science San Jose State University, 2004, 26–56.

Stratonovich RL. Conditional Markov processes. In: *Non-Linear Transformations of Stochastic Processes*. Oxford, UK: Pergamon, 1965, 427–53.

Sutherland GR, Richards RI. Simple tandem DNA repeats and human genetic disease. *Proc Natl Acad Sci USA* 1995;**92**:3636–41.

Tang H, Kirkness EF, Lippert C *et al.* Profiling of short-tandem-repeat disease alleles in 12,632 human whole genomes. *Am J Hum Genet* 2017;**101**:700–15.

Trigiante G, Blanes Ruiz N, Cerase A. Emerging roles of repetitive and repeat-containing RNA in nuclear and chromatin organization and gene expression. *Front Cell Dev Biol* 2021;**9**:735527.

Viterbi A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans Inform Theory* 1967;**13**:260–9.

Wenger AM, Peluso P, Rowell WJ *et al.* Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat Biotechnol* 2019;**37**:1155–62.

Wheeler TJ, Clements J, Eddy SR *et al.* Dfam: a database of repetitive DNA based on profile hidden markov models. *Nucleic Acids Res* 2013;**41**:D70–82.

Willard HF, Waye JS. Hierarchical order in chromosome-specific human alpha satellite DNA. *Trends in Genetics* 1987;**3**:192–8.

Wyman AR, White R. A highly polymorphic locus in human DNA. *Proc Natl Acad Sci USA* 1980;**77**:6754–8.

Yoon B-J, Vaidynathan PP. HMM with auxiliary memory: a new tool for modeling RNA structures. In: *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004*, Vol. 2. IEEE, 2004, 1651–5.

Zattera ML, Bruschi PD. Transposable elements as a source of novel repetitive DNA in the eukaryote genome. *Cells* 2022;**11**:3373.

Zhang H, Li D, Zhao X *et al.* Relatively semi-conservative replication and a folded slippage model for short tandem repeats. *BMC Genomics* 2020;**21**:563–14.

Bioinformatics Advances, 2024, 00, 1–10
https://doi.org/10.1093/bioadv/vbae149
Original Article