Non-parametric and Semi-parametric Support Estimation Using SEquential RESampling Random Walks on Biomolecular Sequences

Wei Wang[1], Jack Smith[1], Hussein A. Hejase[2], and Kevin J. Liu[1(✉)]

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
kjl@msu.edu

[2] Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724, USA

Abstract. Non-parametric and semi-parametric resampling procedures are widely used to perform support estimation in computational biology and bioinformatics. Among the most widely used methods in this class is the standard bootstrap method, which consists of random sampling with replacement. While not requiring assumptions about any particular parametric model for resampling purposes, the bootstrap and related techniques assume that sites are independent and identically distributed (i.i.d.). The i.i.d. assumption can be an over-simplification for many problems in computational biology and bioinformatics. In particular, sequential dependence within biomolecular sequences is often an essential biological feature due to biochemical function, evolutionary processes such as recombination, and other factors.

To relax the simplifying i.i.d. assumption, we propose a new non-parametric/semi-parametric sequential resampling technique that generalizes "Heads-or-Tails" mirrored inputs, a simple but clever technique due to Landan and Graur. The generalized procedure takes the form of random walks along either aligned or unaligned biomolecular sequences. We refer to our new method as the SERES (or "SEquential RESampling") method.

To demonstrate the performance of the new technique, we apply SERES to estimate support for the multiple sequence alignment problem. Using simulated and empirical data, we show that SERES-based support estimation yields comparable or typically better performance compared to state-of-the-art methods.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-00834-5_17) contains supplementary material, which is available to authorized users.

M. Blanchette and A. Ouangraoua (Eds.): RECOMB-CG 2018, LNBI 11183, pp. 294–308, 2018.
https://doi.org/10.1007/978-3-030-00834-5_17

1 Introduction

Resampling methods are widely used throughout computational biology and bioinformatics as a means for assessing statistical support. At a high level, resampling-based support estimation procedures consist of a methodological pipeline: resampled replicates are generated, inference/analysis is performed on each replicate, and results are then compared across replicates. Among the most widely used resampling methods are non-parametric approaches including the standard bootstrap method [5], which consists of random sampling with replacement. We will refer to the standard bootstrap method as the bootstrap method for brevity. Unlike parametric methods, non-parametric approaches need not assume that a particular parametric model is applicable to a problem at hand. However, the bootstrap and other widely used non-parametric approaches assume that observations are independent and identically distributed (i.i.d.).

In the context of biomolecular sequence analysis, there are a variety of biological factors that conflict with this assumption. These include evolutionary processes that cause intra-sequence dependence (e.g., recombination) and functional dependence among biomolecular sequence elements and motifs. Felsenstein presciently noted these limitations when he proposed the application of the bootstrap to phylogenetic inference: "A more serious difficulty is lack of independence of the evolutionary processes in different characters. ... For the purposes of this paper, we will ignore these correlations and assume that they cause no problems; in practice, they pose the most serious challenge to the use of bootstrap methods." (reproduced from p. 785 of [6]).

To relax the simplifying assumption of i.i.d. observations, Landan and Graur [10] introduced the Heads-or-Tails (HoT) technique for the specific problem of multiple sequence alignment (MSA) support estimation. The idea behind HoT is simple but quite powerful: inference/analysis should be repeatable whether an MSA is read either from left-to-right or from right-to-left – i.e., in either heads or tails direction, respectively. While HoT resampling preserves intra-sequence dependence, it is limited to two replicates, which is far fewer than typically needed for reasonable support estimation; often, hundreds of resampled replicates or more are used in practice. Subsequently developed support estimation procedures increased the number of possible replicates by augmenting HoT with bootstrapping, parametric resampling, and domain-specific techniques (e.g., progressive MSA estimation) [11,15,17]. The combined procedures were shown to yield comparable or improved support estimates relative to the original HoT procedure [17] as well as other state-of-the-art parametric and domain-specific methods [9,13], at the cost of some of the generalizability inherent to non-parametric approaches. In this study, we revisit the central question that HoT partially addressed: how can we resample many non-parametric replicates that account for dependence within a sequence of observations, and how can such techniques be used to derive improved support estimates for biomolecular sequence analysis?

2 Methods

In our view, a more general statement of HoT's main insight is the following, which we refer to as the "neighbor preservation property": a neighboring observation is still a neighbor, whether reading an observation sequence from the left or the right. In other words, the key property needed for non-parametric resampling is preservation of neighboring bases within the original sequences, where any pair of bases that appear as neighbors in a resampled sequence must also be neighbors in the corresponding original sequence. To obtain many resampled replicates that account for intra-sequence dependence while retaining the neighbor preservation property, we propose a random walk procedure which generalizes a combination of the bootstrap method and the HoT method. We refer to the new resampling procedure as SERES ("SEquential RESampling"). Note that the neighbor preservation property is necessary but not sufficient for statistical support estimation. Other important properties include computational efficiency of the resampling procedure and unbiased sampling of observations within the original observation sequence.

SERES walks can be performed on both aligned and unaligned sequence inputs. We discuss the case of aligned inputs first, since it is simpler than the case of unaligned inputs.

2.1 SERES Walks on Aligned Sequences

Detailed pseudocode for a non-parametric SERES walk on a fixed MSA is shown in the Appendix's Supplementary Methods section: Algorithm 1.

The random walk is performed on the sequence of aligned characters (i.e., MSA sites). The starting point for the walk is chosen uniformly at random from the alignment sites, and the starting direction is also chosen uniformly at random. The random walk then proceeds in the chosen direction with non-deterministic reversals, or direction changes, that occur with probability γ; furthermore, reversals occur with certainty at the start and end of the fixed MSA. Aligned characters are sampled during each step of the walk. The random walk ends once the number of sampled characters is equal to the fixed MSA length.

The long-term behavior of an infinitely long SERES random walk can be described by a second-order Markov chain. Certain special cases (e.g., $\gamma = 0.5$) can be described using a first-order Markov chain.

In theory, a finite-length SERES random walk can exhibit biased sampling of sites since reversal occurs with certainty at the start and end of the observation sequence, whereas reversal occurs with probability γ elsewhere. However, for practical choices of walk length and reversal probability γ, sampling bias is expected to be minimal.

2.2 SERES Walks on Unaligned Sequences

Detailed pseudocode for SERES resampling of unaligned sequences is shown in the Appendix's Supplementary Methods section: Algorithm 2. Figure 1 provides an illustrated example.

The procedure begins with estimating a set of anchors – sequence regions that exhibit high sequence similarity – which enable resampling synchronization across unaligned sequences. A conservative approach for identifying anchors would be to use highly similar regions that appear in the strict consensus of multiple MSA estimation methods. In practice, we found that highly similar regions within a single guide MSA produced reasonable anchors. We used the average normalized Hamming distance (ANHD) as our similarity measure, where indels are treated as mismatches.

Unaligned sequence indices corresponding to the start and end of each anchor serve as "barriers" in much the same sense as in parallel computing: asynchronous sequence reads occur between barrier pairs along a current direction (left or right), and a random walk is conducted on barrier space in a manner similar to a SERES walk on a sequence of aligned characters. The set of barriers also includes trivial barriers at the start and end of the unaligned sequences. The random walk concludes once the unaligned sequences in the resampled replicate have sufficient length; our criterion requires that the longest resampled sequence has minimum length that is a multiple maxReplicateLengthFactor of the longest input sequence length.

Technically, the anchors in our study make use of parametric MSA estimation and the rest of the SERES walk is non-parametric. The overall procedure is therefore semi-parametric (although see Conclusions for an alternative).

2.3 Performance Study

Our study evaluated the performance of SERES-based support estimation in the context of MSA support estimation. Of course, there are many other applications for non-parametric/semi-parametric support estimation – too many to investigate in one study. We focus on this application since it is considered to be a classical problem in computational biology and bioinformatics and its outputs are useful for studying a range of topics (e.g., phylogenetics and phylogenomics, proteomics, comparative genomics, etc.).

Computational Methods. We examined the problem of evaluating support in the context of MSA estimation. The problem input consists of an estimated MSA A which has a corresponding set of unaligned sequences S. The problem output consists of support estimates for each nucleotide-nucleotide homology in A, where each support estimate is on the unit interval. Note that this computational problem is distinct from the full MSA estimation problem.

There are a variety of existing methods for MSA support estimation. The creators of HoT and their collaborators subsequently developed alignment-specific parametric resampling techniques [11] and then combined the two to obtain two new semi-parametric approaches: GUIDANCE [15] (which we will refer to as GUIDANCE1) and GUIDANCE2 [17]. Other parametric MSA support estimation methods include PSAR [9] and T-Coffee [13].

We focus on GUIDANCE1 and GUIDANCE2, which subsume HoT and have been demonstrated to have comparable or better performance relative to other

state-of-the-art methods [17]. We used MAFFT for re-estimation on resampled replicates, since it has been shown to be among the most accurate progressive MSA methods to date [8,12].

We then used SERES to perform resampling in place of the standard bootstrap that is used in the first step of GUIDANCE1/GUIDANCE2. Re-estimation was performed on 100 SERES replicates – each consisting of a set of unaligned sequences – using MAFFT with default settings, which corresponds to the FFT-NS-2 algorithm for progressive alignment. The SERES resampling procedure used a reversal probability $\gamma = 0.5$, which is equivalent to selecting a direction uniformly at random (UAR) at each step of the random walk; each SERES replicate utilized a total of $\lfloor \frac{k}{20} \rfloor$ anchors with anchor size of 5 bp and a minimum distance between neighboring anchors of 25 bp, where k is the length of the input alignment A. All downstream steps of GUIDANCE1/GUIDANCE2 were then performed using the re-estimated alignments as input.

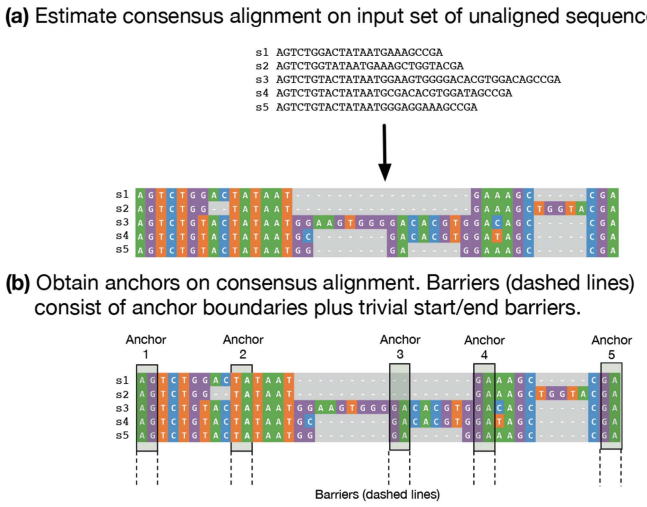

Fig. 1. Illustrated example of SERES resampling random walk on unaligned sequences. Detailed pseudocode is provided in the Appendix's Supplementary Methods section (Algorithm 2 in Appendix). (a) The resampling procedure begins with the estimation of a consensus alignment on the input set of unaligned sequences. (b) A set of conservative anchors is then obtained using the consensus alignment, and anchor boundaries define a set of barriers (including two trivial barriers – one at the start of the sequences and one at the end of the sequences). (c) The SERES random walk is conducted on the set of barriers. The walk begins at a random barrier and proceeds in a random direction to the neighboring barrier. The walk reverses with certainty when the trivial start/end barriers are encountered; furthermore, the walk direction can randomly reverse with probability γ. As the walk proceeds from barrier to barrier, unaligned sequences are sampled between neighboring barrier pairs. (d) The resampling procedure terminates when the resampled sequences meet a specified sequence length threshold.

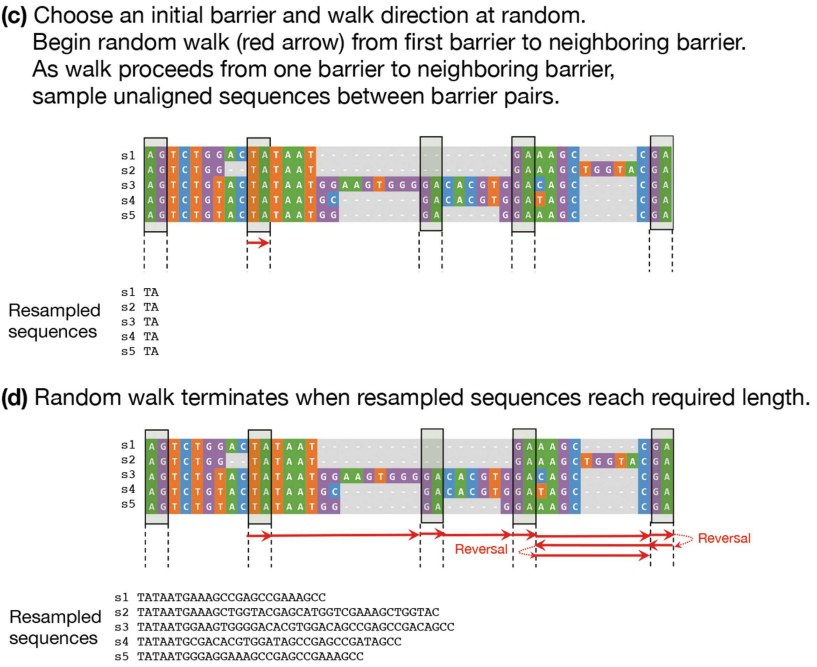

Fig. 1. (*continued*)

Simulated Datasets. Model trees and sequences were simulated using INDELible [7]. First, non-ultrametric model trees with either 10 or 50 taxa were sampled using the following procedure. Model trees were generated under a birth-death process [18], branch lengths were chosen UAR from the interval $(0, 1)$, and the model tree height was re-scaled from its original height h_0 to a desired height h by multiplying all branch lengths by the factor h/h_0. Next, sequences were evolved down each model tree under the General Time-Reversible (GTR) model of substitution [16] and the indel model of Fletcher and Yang [7], where the root sequence had length of 1 kb. We used the substitution rates and base frequencies from the study of Liu et al. [12], which were based upon empirical analysis of the nematode Tree of Life. Sequence insertions/deletions occurred at rate r_i, and we used the medium gap length distribution from the study of Liu et al. [12]. The model parameter values used for simulation are shown in Table 1, and each combination of model parameter values constitutes a model condition. Model conditions are enumerated in order of generally increasing sequence divergence, as reflected by average pairwise ANHD. For each model condition, the simulation procedure was repeated to generate twenty replicate datasets. Summary statistics for simulated datasets are shown in Table 1.

We evaluated performance based upon receiver operating characteristic (ROC) curves, precision-recall curves (PR), and area under ROC and PR curves (ROC-AUC and PR-AUC, respectively). Consistent with other studies of MSA

Table 1. Model condition parameter values and summary statistics. The simulation study parameters consist of the number of taxa, model tree height, and insertion/deletion probability. Each model condition corresponds to a distinct set of model parameter values. The 10-taxon model conditions are named 10.A through 10.E in order of generally increasing sequence divergence; the 50-taxon model conditions are named 50.A through 50.E similarly. The following table columns list average summary statistics for each model condition ($n = 20$). "NHD" is the average normalized Hamming distance of a pair of aligned sequences in the true alignment. "Gappiness" is the percentage of true alignment cells which consists of indels. "True align length" is the length of the true alignment. "Est align length" is the length of the MAFFT-estimated alignment [8] which was provided as input to the support estimation methods. "SP-FN" and "SP-FP" are the proportion of homologies that appear in the true alignment but not in the estimated alignment and vice versa, respectively.

Model condition	Number of taxa	Tree height	Insertion/deletion probability	NHD	Gappiness	True align length	Est align length	SP-FN	SP-FP
10.A	10	0.4	0.13	0.297	0.474	1965.3	1552.3	0.294	0.341
10.B	10	0.7	0.1	0.394	0.512	2165.1	1563.5	0.483	0.533
10.C	10	1	0.06	0.514	0.526	2162.8	1554.0	0.657	0.684
10.D	10	1.6	0.031	0.599	0.485	1874.4	1507.5	0.747	0.752
10.E	10	4.3	0.013	0.693	0.465	1849.3	1612.8	0.945	0.943
50.A	50	0.45	0.06	0.281	0.516	2043.5	1785.7	0.086	0.088
50.B	50	0.7	0.03	0.398	0.475	1935.5	1714.2	0.105	0.102
50.C	50	1	0.02	0.514	0.498	2047.6	1703.1	0.245	0.230
50.D	50	1.8	0.012	0.594	0.471	1945.0	1712.2	0.455	0.419
50.E	50	4.3	0.004	0.688	0.459	1890.2	2319.2	0.963	0.948

support estimation techniques [15,17], the MSA support estimation problem in our study entails annotation of nucleotide-nucleotide homologies in the estimated alignment; thus, homologies that appear in the true alignment but not the estimated alignment are not considered. For this reason, the confusion matrix quantities used for ROC and PR calculations are defined as follows. True positives (TP) are the set of nucleotide-nucleotide homologies that appear in the true alignment and the estimated alignment with support value greater than or equal to a given threshold, false positives (FP) are the set of nucleotide-nucleotide homologies that appear in the estimated alignment with support value greater than or equal to a given threshold but do not appear in the true alignment, false negatives (FN) are the set of nucleotide-nucleotide homologies that appear in the true alignment but appear in the estimated alignment with support value below a given threshold, and true negatives (TN) are the set of nucleotide-nucleotide homologies that do not appear in the true alignment and appear in the estimated alignment with support value below a given threshold. The ROC curve plots the true positive rate ($|TP|/(|TP| + |FN|)$) versus the false positive rate ($|FP|/(|FP| + |TN|)$). The PR curve plots the true positive rate versus precision ($|TP|/(|TP| + |FP|)$). Varying the support threshold yields different points along these curves. Custom scripts were used to perform confusion matrix calculations. ROC curve, PR curve, and AUROC calculations were performed using the scikit-learn Python library [14].

Table 2. Empirical dataset summary statistics. The empirical study made use of reference alignments ("Ref align") from the CRW database [2]. The reference alignments were curated using heterogeneous data including secondary structure information. The column description is identical to Table 1, where the empirical study made use of reference alignments in lieu of the simulation study's true alignments.

Dataset	Number of taxa	NHD	Gappiness	Ref align length	Est align length	SP-FP	SP-FN
IGIA	110	0.606	0.915	10368	6675	0.734	0.784
IGIB	202	0.579	0.910	10633	7379	0.825	0.864
IGIC2	32	0.533	0.700	4243	3514	0.689	0.715
IGID	21	0.719	0.782	5061	3023	0.874	0.904
IGIE	249	0.451	0.838	2751	2775	0.393	0.376
IGIIA	174	0.668	0.814	6406	7005	0.816	0.800
PA23	142	0.293	0.267	3991	3552	0.078	0.077
PE23	117	0.300	0.612	9436	10083	0.202	0.213
PM23	102	0.361	0.797	10999	8803	0.262	0.288
SA16	132	0.212	0.205	1866	1673	0.031	0.028
SA23	144	0.304	0.460	4048	3678	0.077	0.081

Empirical Datasets. We downloaded empirical benchmarks from the Comparative RNA Web (CRW) Site database, which can be found at www.rna.icmb.utexas.edu [2]. In brief, the CRW database includes ribosomal RNA sequence datasets than span a range of dataset sizes and evolutionary divergence. We focused on datasets where high-quality reference alignments are available; the reference alignments were produced using intensive manual curation and analysis of heterogeneous data, including secondary structure information. We selected primary 16 S rRNA, primary 23 S rRNA, primary intron, and seed alignments with at most 250 sequences. Aligned sequences with 99% or more missing data and/or indels were omitted from analysis. Summary statistics for the empirical benchmarks are shown in Table 2.

2.4 Computational Resources Used and Software/Data Availability

All computational analyses were run on computing facilities in Michigan State University's High Performance Computing Center. We used compute nodes in the intel16-k80 cluster, each of which had a 2.4 GHz 14-core Intel Xeon E5-2680v4 processor. All replicates completed with memory usage less than 10 GiB. Open-source software and open data can be found at https://gitlab.msu.edu/liulab/SERES-Scripts-Data.

Table 3. Simulation study results. Results are shown for five 10-taxon model conditions (named 10.A through 10.E in order of generally increasing sequence divergence) and five 50-taxon model conditions (similarly named 50.A through 50.E). We evaluated the performance of two state-of-the-art methods for MSA support estimation – GUIDANCE1 [15] and GUIDANCE2 [17] – versus re-estimation on SERES and parametrically resampled replicates (using parametric techniques from either GUIDANCE1 or GUIDANCE2). (See Methods section for details.) We calculated each method's precision-recall (PR) and receiver operating characteristic (ROC) curves. Performance is evaluated based upon aggregate area under curve (AUC) across all replicates for a model condition ($n = 20$). The top rows show AUC comparisons of GUIDANCE1 ("GUIDANCE1") vs. SERES combined with parametric techniques from GUIDANCE1 ("SERES+GUIDANCE1"), and the bottom rows show AUC comparisons of GUIDANCE2 ("GUIDANCE2") vs. SERES combined with parametric techniques from GUIDANCE2 ("SERES+GUIDANCE2"); for each model condition and pairwise comparison, the best AUC is shown in bold. Statistical significance of PR-AUC or AUC-ROC differences was assessed using a one-tailed pairwise t-test or DeLong et al. [4] test, respectively, and multiple test correction was performed using the method of Benjamini and Hochberg [1]. Corrected q-values are reported ($n = 20$) and all were significant ($\alpha = 0.05$).

Model condition	PR-AUC (%)		Pairwise t-test corrected q-value	ROC-AUC (%)		DeLong et al. test corrected q-value
	GUIDANCE1	SERES+GUIDANCE1		GUIDANCE1	SERES+GUIDANCE1	
10.A	88.74	**91.17**	5.4×10^{-7}	80.22	**85.57**	$< 10^{-10}$
10.B	82.21	**86.26**	1.5×10^{-6}	84.83	**88.66**	$< 10^{-10}$
10.C	76.23	**83.49**	1.9×10^{-4}	86.98	**91.23**	$< 10^{-10}$
10.D	74.65	**85.81**	1.9×10^{-4}	88.55	**93.72**	$< 10^{-10}$
10.E	42.61	**59.20**	3.1×10^{-4}	82.24	**87.40**	$< 10^{-10}$
50.A	98.22	**98.92**	5.3×10^{-10}	83.09	**90.64**	$< 10^{-10}$
50.B	97.84	**98.69**	2.8×10^{-9}	82.85	**90.39**	$< 10^{-10}$
50.C	95.08	**96.80**	5.6×10^{-8}	85.54	**90.64**	$< 10^{-10}$
50.D	90.79	**95.75**	5.3×10^{-6}	88.89	**94.56**	$< 10^{-10}$
50.E	62.47	**79.14**	8.0×10^{-10}	91.02	**93.23**	$< 10^{-10}$
Model condition	PR-AUC (%)		Pairwise t-test corrected q-value	ROC-AOC (%)		DeLong et al. test corrected q-value
	GUIDANCE2	SERES+GUIDANCE2		GUIDANCE2	SERES+GUIDANCE2	
10.A	92.55	**93.33**	7.4×10^{-6}	87.17	**88.34**	$< 10^{-10}$
10.B	88.08	**89.31**	8.4×10^{-4}	89.45	**90.56**	$< 10^{-10}$
10.C	84.28	**86.86**	3.1×10^{-4}	91.36	**92.88**	$< 10^{-10}$
10.D	86.03	**88.75**	1.9×10^{-4}	93.34	**94.69**	$< 10^{-10}$
10.E	51.17	**62.30**	1.3×10^{-3}	86.00	**88.28**	$< 10^{-10}$
50.A	98.98	**99.14**	5.3×10^{-6}	91.17	**92.50**	$< 10^{-10}$
50.B	98.79	**98.96**	1.5×10^{-6}	91.24	**92.44**	$< 10^{-10}$
50.C	96.86	**97.45**	3.2×10^{-7}	90.81	**92.31**	$< 10^{-10}$
50.D	94.04	**96.23**	1.5×10^{-5}	92.67	**95.09**	$< 10^{-10}$
50.E	72.61	**81.47**	1.5×10^{-8}	92.94	**94.22**	$< 10^{-10}$

3 Results

3.1 Simulation Study

For all model conditions, SERES-based resampling and re-estimation yielded improved MSA support estimates compared to GUIDANCE1 and GUIDANCE2, two state-of-the-art methods, where performance was measured by PR-AUC or ROC-AUC (Table 3). In all cases, PR-AUC or ROC-AUC improvements were statistically significant (corrected pairwise t-test or DeLong et al. [4] test, respectively; $n = 20$ and $\alpha = 0.05$). The observed performance improvement was robust to several experimental factors: dataset size, increasing sequence divergence due to increasing numbers of substitutions, insertions, and deletions, and the choice of alignment-specific parametric support estimation techniques (i.e., the parametric approaches used by either GUIDANCE1 or GUIDANCE2) that were used in combination with SERES-based support estimation.

Compared to dataset size, sequence divergence had a relatively greater quantitative impact on each method's performance. For each dataset size (10 or 50 taxa), PR-AUC differed by at most 3% on the least divergent model condition. The SERES-based method's performance advantage grew as sequence divergence increased – to as much as 28% – and the largest performance advantages were seen on the most divergent datasets in our study. The most divergent datasets were also the most challenging. For each method, PR-AUC generally degraded as sequence divergence increased; however, the SERES-based method's PR-AUC degraded more slowly compared to the non-SERES-based method. Consistent with the study of Sela et al. [17], GUIDANCE2 consistently outperformed GUIDANCE1 on each model conditions and using either AUC measure. The performance improvement of SERES+GUIDANCE1 over GUIDANCE1 was generally greater than that seen when comparing SERES+GUIDANCE2 and GUIDANCE2; furthermore, the PR-AUC-based corrected q-values were more significant for the former compared to the latter in all cases except for the 10.D model condition, where the corrected q-values were comparable. Finally, while the SERES-based method consistently yielded performance improvements over the corresponding non-SERES-based method regardless of the choice of performance measure (either PR-AUC or ROC-AUC), the PR-AUC difference was generally larger than the ROC-AUC difference, especially on more divergent model conditions. On average across all replicates of all model conditions with a given dataset size, the runtime overhead contributed by SERES was minimal – amounting to just a few minutes per replicate dataset – and all methods in the simulation study completed analysis of each replicate dataset in less than half an hour (Supplementary Table S1 in Appendix).

3.2 Empirical Study

Relative to GUIDANCE1 or GUIDANCE2, SERES-based support estimates consistently returned higher AUC on all datasets – primary, seed, and intronic – with a single exception: the comparison of SERES+GUIDANCE2 and

Table 4. Empirical study results. The empirical study made use of benchmark RNA datasets and curated reference alignments from the CRW database [2]. Results are shown for intronic ("IG" prefix) and non-intronic datasets ("P" prefix and "S" prefix, following "primary" and "seed" nomenclature from the CRW database). For each dataset, we report each method's PR-AUC and ROC-AUC. For each dataset and pairwise method comparison, the best AUC is shown in bold. Methods, performance measures, table layout, and table description are otherwise identical to Table 3.

Dataset	PR-AUC (%)		ROC-AUC (%)	
	GUIDANCE1	SERES+ GUIDANCE1	GUIDANCE1	SERES+ GUIDANCE1
IGIA	62.67	**69.28**	89.50	**91.62**
IGIB	73.60	**87.47**	94.49	**97.39**
IGIC2	72.67	**75.36**	82.25	**83.87**
IGID	63.74	**76.30**	95.10	**96.73**
IGIE	93.56	**95.42**	90.08	**93.30**
IGIIA	73.03	**83.06**	86.49	**96.45**
PA23	98.54	**99.41**	82.59	**93.63**
PE23	98.44	**99.27**	94.75	**97.41**
PM23	97.53	**98.48**	94.20	**96.44**
SA16	99.72	**99.86**	91.07	**95.57**
SA23	98.35	**99.24**	81.76	**92.18**
Dataset	PR-AUC (%)		ROC-AUC (%)	
	GUIDANCE2	SERES+ GUIDANCE2	GUIDANCE2	SERES+ GUIDANCE2
IGIA	67.4	**68.49**	91.38	**91.94**
IGIB	80.66	**86.72**	96.47	**97.38**
IGIC2	**74.44**	73.27	**84.63**	82.51
IGID	75.15	**78.38**	96.44	**97.09**
IGIE	94.6	**95.44**	91.84	**93.49**
IGIIA	78.16	**85.09**	94.50	**96.82**
PA23	99.24	**99.53**	91.48	**94.88**
PE23	99.07	**99.34**	96.72	**97.63**
PM23	98.68	**98.85**	96.93	**97.28**
SA16	99.88	**99.91**	96.22	**97.22**
SA23	99.04	**99.33**	89.93	**93.18**

GUIDANCE2 on the intronic IGIC2 dataset, where the PR-AUC and ROC-AUC differences were 1.17% and 2.12%, respectively. For each pairwise comparison of methods (i.e., SERES+GUIDANCE1 vs. GUIDANCE1 or SERES+GUIDANCE2 vs. GUIDANCE2), the SERES-based method returned relatively larger PR-AUC improvements on datasets with greater sequence

divergence, as measured by ANHD and gappiness. In particular, PR-AUC improvements were less than 1% on seed and primary non-intronic datasets. Intronic datasets yielded PR-AUC improvements of as much as 13.87%. Observed AUC improvements of SERES+GUIDANCE1 over GUIDANCE1 were relatively greater than those seen for SERES+GUIDANCE2 in comparison to GUIDANCE2. Finally, GUIDANCE2 consistently returned higher AUC relative to GUIDANCE1, regardless of whether PR or ROC curves were the basis for AUC comparison.

4 Discussion

Re-estimation using SERES resampling resulted in comparable or typically improved support estimates for the applications in our study. We believe that this performance advantage is due to the ability to generate many distinct replicates while enforcing the neighbor preservation principle. The latter is critical for retaining sequence dependence which is inherent to the application in our study.

On all model conditions, SERES+GUIDANCE1 support estimation resulted in significant improvements in AUC-PR and AUC-ROC compared to GUIDANCE1. A similar outcome was observed when comparing SERES+GUIDANCE2 and GUIDANCE2. The main difference in each comparison is the resampling technique – either SERES or standard bootstrap. Our findings clearly demonstrate the performance advantage of the former over the latter. SERES accounts for intra-sequence dependence due to insertion and deletion processes, while the bootstrap method assumes that sites are independent and identically distributed. Regarding comparisons involving GUIDANCE2 versus GUIDANCE1, a contributing factor may have been the greater AUC of GUIDANCE2 over GUIDANCE1. We used SERES to perform semi-parametric support estimation in conjunction with the parametric support techniques of GUIDANCE1 or GUIDANCE2. The latter method's relatively greater AUC may be more challenging to improve upon (Table 4).

The performance comparisons on empirical benchmarks were consistent with the simulation study. In terms of ANHD and gappiness, the non-intronic datasets in our empirical study were more like the low divergence model conditions in our simulation study, and the intronic datasets were more like the higher divergence model conditions. Across all empirical datasets, SERES-based support estimation consistently yielded comparable or better AUC versus GUIDANCE1 or GUIDANCE2 alone. The SERES-based method's AUC advantage generally increased as datasets became more divergent and challenging to align – particularly when comparing performance on non-intronic versus intronic datasets. We found that the support estimation methods returned comparable AUC (within a few percentage points) on datasets with 1–2 dozen sequences and low sequence divergence relative to other datasets. In particular, IGIC2 was the only dataset where SERES+GUIDANCE2 did not return an improved AUC relative to GUIDANCE2. IGIC2 was the second-smallest dataset – about an order

of magnitude smaller than all other datasets except the IGID dataset – and IGIC2 also had the second-lowest ANHD and lowest gappiness among intronic datasets. IGID was the smallest dataset, but had higher ANHD and gappiness compared to the IGIC2 dataset. Compared to the other empirical datasets, SERES+GUIDANCE2 returned a small AUC improvement over GUIDANCE2 on the IGID dataset – at most 3.2%.

On simulated and empirical datasets, greater sequence divergence generally resulted in increased inference error for all methods. However, the SERES-based method's performance tended to degrade more slowly than the corresponding non-SERES-based method as sequence divergence increased, and the greatest performance advantage was seen on the most divergent model conditions and empirical datasets.

Finally, we note that non-parametric/semi-parametric resampling techniques are orthogonal to parametric alternatives. Consistent with previous studies [15, 17], we found that combining two different classes of methods yielded better performance than either by itself.

5 Conclusions

This study introduced SERES, which consists of new non-parametric and semi-parametric techniques for resampling biomolecular sequence data. Using simulated and empirical data, we explored the use of SERES resampling for support estimation involving a classical problem in computational biology and bioinformatics. We found that SERES-based support estimation yields comparable or typically better performance compared to state-of-the-art approaches.

We conclude with possible directions for future work. First, the SERES algorithm in our study made use of a semi-parametric resampling procedure on unaligned inputs, since anchors were constructed using progressive multiple sequence alignment. While this approach worked well in our experiments, non-parametric alternatives could be substituted (e.g., unsupervised k-mer clustering using alignment-free distances [3]) to obtain a purely non-parametric resampling procedure. Second, the unaligned input application focused on nucleotide-nucleotide homologies to enable direct comparison against existing MSA support estimation procedures (i.e., GUIDANCE1 and GUIDANCE2). The SERES framework can be extended in a straightforward manner to estimate support for nucleotide-indel pairs. Third, SERES resampling can be used to perform full MSA inference. One approach would be to analyze homologies that appeared in re-estimated inferences across resampled replicates, without regard to any input alignment. Fourth, in the case where biomolecular sequences evolved under insertion/deletion processes, we consider the distinction between aligned and unaligned inputs to be an unnecessary dichotomy. In theory, the latter subsumes the former. We can apply this insight using a two-phase approach: (1) perform SERES-based re-estimation on unaligned sequences to estimate support for aligned homologies (from either an input MSA or the de novo procedure proposed above), and (2) perform support-weighted SERES walks on the annotated MSA from the previous stage to obtain support estimates on downstream

inference. Alternatively, we can simultaneously address both problems using co-estimation. Finally, we envision many other SERES applications. Examples in computational biology and bioinformatics include protein structure prediction, detecting genomic patterns of natural selection, and read mapping and assembly. Non-parametric resampling for support estimation is widely used throughout science and engineering, and SERES resampling may similarly prove useful in research areas outside of computational biology and bioinformatics.

Acknowledgments. This work has been supported in part by the National Science Foundation (grant nos. CCF-1565719, CCF-1714417, and DEB-1737898 to KJL) and MSU faculty startup funds (to KJL). Computational experiments were performed using the High Performance Computing Center (HPCC) at MSU.

References

1. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. Royal Stat. Soc. Ser. B (Methodol) **57**(1), 289–300 (1995)
2. Cannone, J.J., et al.: The Comparative RNA Web (CRW) site: an online database of comparative sequence and structure information for Ribosomal, Intron and Other RNAs. BMC Bioinform. **3**(15) (2002). http://www.rna.ccbb.utexas.edu
3. Daskalakis, C., Roch, S.: Alignment-free phylogenetic reconstruction. In: Berger, B. (ed.) RECOMB 2010. LNCS, vol. 6044, pp. 123–137. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12683-3_9
4. DeLong, E.R., DeLong, D.M., Clarke-Pearson, D.L.: Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics **44**(3), 837–845 (1988)
5. Efron, B.: Bootstrap methods: another look at the jackknife. Ann. Stat. **7**(1), 1–26 (1979)
6. Felsenstein, J.: Confidence limits on phylogenies: an approach using the bootstrap. Evolution **39**(4), 783–791 (1985)
7. Fletcher, W., Yang, Z.: INDELible: a flexible simulator of biological sequence evolution. Mol. Biol. Evol. **26**(8), 1879–1888 (2009)
8. Katoh, K., Standley, D.M., Kazutaka Katoh and Daron: MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol. Biol. Evol. **30**(4), 772–780 (2013)
9. Kim, J., Ma, J.: PSAR: measuring multiple sequence alignment reliability by probabilistic sampling. Nucleic Acids Res. **39**(15), 6359–6368 (2011)
10. Landan, G., Graur, D.: Heads or tails: a simple reliability check for multiple sequence alignments. Mol. Biol. Evol. **24**(6), 1380–1383 (2007)
11. Landan, G., Graur, D.: Local reliability measures from sets of co-optimal multiple sequence alignments. In: Biocomputing, pp. 15–24. World Scientific (2008)
12. Liu, K., et al.: SATé-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Syst. Biol. **61**(1), 90–106 (2012)
13. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: a novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. **302**, 205–217 (2000)
14. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

15. Penn, O., Privman, E., Landan, G., Graur, D., Pupko, T.: An alignment confidence score capturing robustness to guide tree uncertainty. Mol. Biol. Evol. **27**(8), 1759–1767 (2010)
16. Rodriguez, F., Oliver, J.L., Marin, A., Medina, J.R.: The general stochastic model of nucleotide substitution. J. Theor. Biol. **142**, 485–501 (1990)
17. Sela, I., Ashkenazy, H., Katoh, K., Pupko, T.: GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters. Nucleic Acids Res. **43**(W1), W7–W14 (2015)
18. Yang, Z., Rannala, B.: Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method. Mol. Biol. Evol. **14**(7), 717–724 (1997)

Appendix: non-parametric and semi-parametric support estimation using SEquential RESampling random walks on biomolecular sequences

Wei Wang[1], Jack Smith[1], Hussein A. Hejase[2], and Kevin J. Liu[1]

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
[2] Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724, USA
Correspondence: kjl@msu.edu

1 Supplementary Methods

1.1 SERES walks on aligned sequences

The pseudocode for a non-parametric SERES walk on a fixed MSA A is shown in Algorithm 1.

Algorithm 1 SERES walk on aligned sequences

```
1: procedure SERESWalkOnAlignedSequences(A, γ, numReplicates)
       ▷ Input: MSA A, walk reversal probability γ, number of SERES replicates numReplicates
                                                          ▷ Output: list of SERES replicates
2:     replicates = <>
3:     for i = 1 to numReplicates do
4:         direction = (rand() > 0.5) ? +1 : −1    ▷ Uniformly at random (UAR) choose direction (right vs. left)
5:         i = ⌊ length(A) * rand() ⌋ + 1                          ▷ UAR draw from [1, length(A)]
                                  ▷ rand() returns floating point number sampled UAR from [0, 1)
6:         replicate = <>
7:         while length(replicate) < length(A) do
8:             replicate .= A_i                  ▷ read A_i, which is the ith character in alignment A
                                                  ▷ Alignment characters A_i are one-indexed
9:             i+= direction
10:            if (i ≤ 0) or (i > length(A)) or (rand() < γ) then
                                                                  ▷ Reflection of random walk
11:                direction *= -1
12:                if (i ≤ 0) or (i > length(A)) then
13:                    i+= direction * 2          ▷ Always reflect at start/end of alignment A
14:        replicates .= replicate
15:    return(replicates)
```

1.2 SERES walks on unaligned sequences

The pseudocode for SERES resampling of a set of unaligned sequences S is shown in Algorithms 2 through 4.

Algorithm 2 SERES resampling of unaligned sequences

procedure SERESWalkOnUnalignedSequences(S, γ, numReplicates)
▷ Input: set of unaligned sequences S, walk reversal probability γ, number of SERES replicates numReplicates
▷ Output: list of SERES replicates
    replicates = <>
    barriers = <>
    A_{init} = ObtainGuideAlignments(S) ▷ See Algorithm 3
    Ψ = GetAnchorsFromGuideAlignments(S, A_{init}) ▷ See Algorithm 3
    AddTrivialBarriers(barriers)
    for each ($\boldsymbol{a}$, $\boldsymbol{b}$) Ψ **do**
        barriers .= $\boldsymbol{a}$. $\boldsymbol{b}$
    for i = 1 to numReplicates **do**
        replicates .= SERESWalkOnUnalignedSequences(S, γ, i, barriers)
    return(replicates)

static variable maxReplicateLengthFactor ▷ Maximum replicate length is factor of longest unaligned sequence length
procedure SERESWalkOnUnalignedSequences(S, γ, replicateNum, barriers)
    direction = (rand() > 0.5) ? +1 : −1 ▷ UAR choose direction (left vs. right)
    i = ⌊ length(barriers) * rand() ⌋ + 1
    replicate = <>
    while maxLength(replicate) < maxLength(S) * maxReplicateLengthFactor **do** ▷ maxLength(S) is length of longest unaligned sequence in S
        if ((i == 1) **and** (direction == −1)) **or** ((i == length(barriers)) **and** (direction == +1)) **then** ▷ reflect at first or last barrier
            direction *= −1
        AsynchronousReadBetweenAdjacentBarriers(S, barriers, i, direction, replicate) ▷ read result passed by reference to mutable object replicate
        i += direction
        if rand() < γ **then** ▷ change walk direction with probability γ
            direction *= −1
    return(replicate)

procedure AsynchronousReadBetweenAdjacentBarriers(S, barriers, i, direction, replicate)
    j = i + direction
    for z = i to n **do**
        replicate[z] .= (direction > 0) ? substr(S[z], barriers[i], barriers[j]) : reverse(substr(S[z], barriers[j] + 1, barriers[i] + 1)
        ▷ substr(x, i, j) returns substring in index interval $[i, j)$ if $i < j$ or empty string if $i \geq j$
    return ▷ read result passed by reference to mutable object replicate

Algorithm 3 Obtain anchors

```
1: static variable M                                   ▷ MSA methods M =< M1, M2, ... >
2: procedure ObtainGuideAlignments(S)
3:     alignments = <>
4:     for each (m) M do
5:         alignments .= m(S)
6:     return(alignments)

7: procedure GetAnchorsFromGuideAlignments(S, A_init)
8:     α = <>
9:     β = <>
10:    canonicalAlignment = A_init[1]   ▷ anchors are indexed based on a fixed alignment in A_init
       (WLOG chosen to be the first alignment in A_init)
11:    C_strict = GetStrictConsensusColumns(A_init)   ▷ GetStrictConsensusColumns() returns
       column indices into first alignment in canonicalAlignment
12:    α_strict = MergeAdjacentColumns(A_init, C_strict)   ▷ merges adjacent columns
       ▷ returns array of ordered pairs (x, y) where start indices x and end indices y are indexed
       based on canonicalAlignment
13:    SortAnchors(α_strict, canonicalAlignment)
14:    for z = 1 to length(α_strict) do
15:        for i = 1 to n do
16:            (x, y) = α_strict[z]
17:            if substr(canonicalAlignment[i], x[i], y[i]) contains only indels then
18:                α[i][z] = LookupUnalignedSequenceIndex(
       GetLastNonIndelIndexInPrefix(canonicalAlignment[i], x[i]))
19:                β[i][z] = α[i][z]
20:            else
21:                α[i][z] = LookupUnalignedSequenceIndex(
       GetFirstNonIndelIndexInRange(canonicalAlignment[i], x[i], y[i] + 1))
22:                β[i][z] = LookupUnalignedSequenceIndex(
       GetLastNonIndelIndexInRange(canonicalAlignment[i], x[i], y[i] + 1))
23:    return(α, β)

24: procedure SortAnchors(α, canonicalAlignment)
       ▷ α is an array of ordered pairs (x, y) where start indices x and end indices y are indexed
       based on canonicalAlignment
25:    sort (ComputeModifiedHammingDistance(u, canonicalAlignment) <=>
       ComputeModifiedHammingDistance(v, canonicalAlignment)) α
                                                   ▷ perl sort syntax
                                                   ▷ See Algorithm 4
```

Algorithm 4 Modified Hamming distance calculation

```
1: procedure ComputeModifiedHammingDistance(u, A)
2:     dist = 0
3:     (x, y) = u
4:     for i = 1 to n do
5:         for j = i + 1 to n do
6:             dist += ComputeModifiedHammingDistancePair(substr(A[i],x[i],y[i]),
       substr(A[j],x[j],y[j]))
7:     return(dist / (n choose 2))

8: procedure ComputeModifiedHammingDistancePair(x, y)
9:     alignedLength = length(x)                ▷ aligned sequences x and y have same length
10:    matches = 0
11:    for i = 1 to alignedLength do
12:        if (x[i] != INDEL) and (y[i] != INDEL) and (x[i] == y[i]) then
                                   ▷ homologies involving indels are penalized as mismatch
13:            matches++
14:    return(matches / alignedLength)
```

2 Supplementary Results

Runtime information for simulation study experiments are shown in Supplementary Table S1.

Supplementary Table S1. Method runtime in simulation study experiments. Method runtime in seconds is reported as an average across all replicates of all model conditions with a given dataset size – either 10 or 50 taxa ($n = 100$).

Model condition class (based on number of taxa)	Method	Runtime (s)
10	GUIDANCE1	62.52
10	SERES+GUIDANCE1	105.60
10	GUIDANCE2	218.16
10	SERES+GUIDANCE2	326.19
50	GUIDANCE1	523.49
50	SERES+GUIDANCE1	760.75
50	GUIDANCE2	1025.83
50	SERES+GUIDANCE2	1318.27