Automated Removal of Non-homologous Sequence Stretches with PREQUAL

Iker Irisarri, Fabien Burki, and Simon Whelan

Abstract

Large-scale multigene datasets used in phylogenomics and comparative genomics often contain sequence errors inherited from source genomes and transcriptomes. These errors typically manifest as stretches of non-homologous characters and derive from sequencing, assembly, and/or annotation errors. The lack of automatic tools to detect and remove sequence errors leads to the propagation of these errors in large-scale datasets. PREQUAL is a command line tool that identifies and masks regions with non-homologous adjacent characters in sets of unaligned homologous sequences. PREQUAL uses a full probabilistic approach based on pair hidden Markov models. On the front end, PREQUAL is user-friendly and simple to use while also allowing full customization to adjust filtering sensitivity. It is primarily aimed at amino acid sequences but can handle protein-coding nucleotide sequences. PREQUAL is computationally efficient and shows high sensitivity and accuracy. In this chapter, we briefly introduce the motivation for PREQUAL and its underlying methodology, followed by a description of basic and advanced usage, and conclude with some notes and recommendations. PREQUAL fills an important gap in the current bioinformatics tool kit for phylogenomics, contributing toward increased accuracy and reproducibility in future studies.

Key words Filtering, Genomics, HMM, Homology, Phylogenomics, Sequence analysis

1 Introduction

The assembly of large datasets used in phylogenomic and comparative genomic analyses heavily relies on automatic tools, typically run sequentially into a pipeline. There are, however, few automatic tools for quality control, resulting in these important controls being performed (semi-) manually or worse, ignored. One such control is the removal of primary sequence errors inherited from source genomes and transcriptomes. These errors typically arise from sequencing or assembly errors that insert (or delete) a nucleotide, leading to the disruption of the reading frame in translated protein-coding genes. With poor sequence quality, it is not uncommon that the reading frame is re-established by a second downstream deletion (or insertion), producing stretches of

non-homologous amino acids within an otherwise legitimate sequence. Another common situation is represented by poorly annotated eukaryotic genes where intron-exon boundaries are incorrectly predicted, resulting in translated intronic regions being forced into multiple sequence alignments (MSAs).

The impact of these sequence errors in downstream analyses is not yet fully understood, but it is reasonable to assume that they will adversely affect phylogenetic analyses. Persistence of non-homologous sequence stretches will negatively affect the quality of MSA, which is known to impact phylogenetic accuracy [1]. The non-homologous sequence itself might also cause problems. If the phylogenetic signal in the source data is weak, then errors induced by the primary sequence error or its effect on the MSA might lead to false groupings in the tree, akin to the systematic errors caused by model misspecifications [2, 3]. Furthermore, primary sequence errors can bias the estimation of natural selection coefficients (dN/dS) in a similar manner to errors in MSA, because they lead to an artificially inflated number of non-synonymous substitutions [4]. Di Franco et al. [5] have found that segment filtering leads to improved branch length estimation and a lower false-positive rate in detecting positive selection.

When we look at MSAs, primary sequence errors are typically evident as stretches of residues that share no sitewise homology with other sequences in otherwise evolutionarily conserved regions (*see* Fig. 1a). These stretches might be identified in a labor-intensive and qualitative way through looking at MSAs with homologous sequences from other species, but fixing this problem is at best painstaking for a phylogenomic study with hundreds of genes and species. For this reason, this step has been ignored by most phylogenomic studies so far. Even after that hard work, the result is based on individual opinion, making it subjective and not reproducible.

To address this problem we developed PREQUAL (PRE-alignment QUALity filter; [6]): a command line tool that, given a set of unaligned homologous sequences as input, can identify and mask regions with non-homologous adjacent characters. Our approach (*see* Fig. 1b) is different from the standard trimming of MSAs to remove poorly aligned columns (*see* Fig. 1c), for which several tools exist, including BMGE [7], trimAl [8], Gblocks [9], and Divvier [10]. We deal with non-homologous residues in individual sequences that are not necessarily associated with regions of poor alignment. In fact, PREQUAL uses unaligned sets of sequences and thus does not assume any fixed sequence alignment. Instead of simply removing these errors, PREQUAL masks them in order to maintain the original distance between residues, thereby facilitating the inference of positional homology in the MSA step.

A Original alignment

B PREQUAL

C PREQUAL + BMGE

Fig. 1 Primary sequence errors and performance of stretch-masking vs. block filtering. **(a)** Alignment of Hedgehog-interacted protein (HHIP) orthologs from vertebrate genomes (OrthoDB acc. EOG090702V5; <http://www.orthodb.org/>; accessed 07/11/2017). **(b)** PREQUAL masks non-homologous residue stretches in individual sequences with X (light blue). Note that PREQUAL works on unaligned sequences but here aligned for easy identification of masked residues. **(c)** Alignment block filtering with BMGE on PREQUAL-treated and aligned sequences. Images produced with AliView [20]

Moreover, masking likely non-homologous stretches should facilitate the inference of MSAs, thereby improving MSA quality and reducing the known biases in downstream phylogenomic and comparative genomic analyses.

2 Methods

2.1 Methodological Intuition

Before discussing the technical details of what PREQUAL does, we will give a broad overview of what it is trying to achieve and how it works. We can formulate the problem of non-homologous residues as a hypothesis for each individual residue in every sequence stating “is there evidence that this residue shares sitewise homology with another residue in my data set?” In other words, if I were to consider all the probable ways by which a residue could have evolved, is there support that this residue shares ancestry with any other residue in my data? If the answer is yes, then that residue should be carried forward to future steps (although we note it could be removed later for other reasons, such as poor alignment quality). If the answer is no, then that residue should be excluded

from future steps, either through masking to maintain the relative positions of the residues or through simple removal.

With this hypothesis in mind, the next question is how to quantify the support for homology and pick a threshold that specifies yes or no. Our quantification works by comparing a sequence with every other sequence in our data, and for each residue in that sequence, we calculate the maximal probability of that residue sharing sitewise homology with any other residue in the data using an explicit probability model. This gives a score for every residue in every sequence, which is compared to an a priori chosen threshold to decide whether to keep or remove the residue. In order to derive an efficient threshold, we took both a simulation approach—where we introduced errors to error-free simulated sequences and found the best score to identify those errors—and an empirical approach looking at real data and assessing whether the removed residues showed signs of errors. Our results showed that both approaches yielded very similar thresholds, giving us confidence that PREQUAL works as intended [6].

This intuition is also helpful for explaining why PREQUAL is designed to work for amino acid sequences (or translated to amino acids from protein-coding DNA), but not straight on DNA sequences. In order to test our hypothesis of sitewise homology, we need sequences to be relatively complex, so that we can reliably distinguish true homologies from errors. Amino acid sequences have 20 possible characters and are therefore much more complex than DNA sequences that have only four. To put this in numbers, consider a sequence of length 10: there are around one million possible DNA sequences ($4^{10} = 1,048,576$) but around 10 trillion possible amino acid sequences ($20^{10} = 10.24 \times 10^{12}$). The vastly greater space of possible amino acid sequences makes identifying similarities—and therefore errors—much easier.

2.2 Statistical Approach

PREQUAL uses a full probabilistic approach based on pair hidden Markov models (pairHMM) to describe the relationship between sequence pairs. Given a parameterized pairHMM, it calculates the posterior probability (PP) of a character being related to a character from another sequence and filters out (masks) characters with insufficient evidence of homology. PairHMMs consist of three hidden states defining the relationship between two sequences X and Y: a match state that describes shared ancestry between X and Y through a process of substitution and insert and delete states that describe gain and loss in sequence X, respectively. Given a parameterized pairHMM, it is possible to calculate the PP of a character from X being related to a character from Y using the forward-backward algorithm [11]. In PREQUAL, a slightly different PP is calculated to capture the PP of a character in sequence $X = \{x_i\}$ sharing a common ancestor with any character in the set of n sequences being considered $\mathbf{A} = \{Y^1, \dots, Y^n\}$, where $X \in \mathbf{A}$; the

set of sequences without X is $\mathbf{A}' = \mathbf{A} - X$, and a pairHMM can be run on the pair (X, Y) to obtain the posterior probability of $\Pr(x_i, y_j)$ using forward-backward. First, we can calculate the maximal PP of x_i sharing a common ancestor with any character in Y as $\Pr(x_i, Y_*) = \max_{y_j \in Y} \{ \Pr(x_i, y_j) \}$. Then we want to find the maximal PP of x_i sharing a common ancestor with any of the other sequences: $\Pr(x_i | \text{Ancestor}) = \max_{Y \in \mathbf{A}'} \{ \Pr(x_i, Y_*) \}$. The value of $\Pr(x_i | \text{Ancestor})$ can be computed for every character of every sequence in $\mathbf{A}$, and an appropriate threshold τ can be used as a cutoff to discriminate between characters with enough evidence for shared ancestry that should be carried through to the alignment phase and those that should be discarded (masked).

In practice, PREQUAL calculates PPs using a bounded pairHMM with a substantially modified version of Zorro [12]. It uses a heuristic approach for calculating $\Pr(x_i | \text{Ancestor})$ by choosing a set of sequences from $\mathbf{S}'$ based on evolutionary divergence [13] and sequence coverage to reduce the number of pairHMM calculations that need to be performed. This heuristic samples only a subset of sequences to find the $\max(\Pr(x_i | \text{Ancestor}))$ ensuring that the closest sequences are included and they have adequate similarity over enough of the characters. The default is to examine the ten closest sequences ensuring that at least three of them are equal to or longer than the median length of all the sequences. PREQUAL also avoids computing the entire dynamic programming matrix for the pairHMM where possible, by bounding the distance any path can take through that matrix from the diagonal.

The residue-specific PP is the main filter in PREQUAL. The default threshold is $\text{PP} = 0.994$, which has been derived from a ROC curve so that $\geq 95\%$ of correct amino acids are retained while removing $> 90\%$ of the errors (*see* Fig. 4). The performance of the default threshold has also been validated on various real phylogenomic datasets (*see* Subheading 4).

3 Program Usage

PREQUAL is designed to be as simple and lightweight to use as possible. The default values should work well for most phylogenomic datasets. Nevertheless, PREQUAL provides the user with many options for customizing its behavior and comes with a detailed manual.

3.1 Download and Installation

The program PREQUAL is written in C/C++, and it is available through a GNU GPL v3.0 from Simon Whelan's GitHub (<https://github.com/simonwhelan/prequal>).

PREQUAL can be built from source with a simple “make” command in Linux and OS X. After installation, add it to your path if you want PREQUAL to be accessible from anywhere in your system.

3.2 Basic Usage

PREQUAL runs on the command line. The majority of users will find the basic usage sufficient (obtained by typing the program name without any argument). Given an input file containing a set of homologous sequences (in FASTA format) and assuming that PREQUAL is in the current directory, type:

```
./prequal my_seq.fasta
```

This command will run PREQUAL on the input file. The input file should be a set of homologous sequences in FASTA format and must not contain stop codons (*).

The default output of PREQUAL consists of at least two files:

`my_seq.fasta.filtered`: main output containing the sequences after filtering.

`my_seq.fasta.PP`: internal file containing the calculated PP. The interest of saving the PPs is to rerun PREQUAL with different settings on the same data without the need of recalculating PPs, the most computationally intensive step (*see Note 1*).

`my_seq.fasta.warning`: this file might be also generated, containing warnings when too much of a sequence (or all of it) has been filtered out.

Figure 2 shows a typical output printed to the screen. It can be separated into the following sections:

A Summary of input data.

B Information about the computation of PPs. By default, PREQUAL’s heuristic will compare each sequence to the ten most similar sequences given adequate coverage (B1). Progress spinners provide an indication of the progress (B2). A confirmation will be printed when PPs are calculated (B3).

C This section refers to the actual filtering step. It provides information on the applied PP threshold and the number of residues that fall below it and thus will be removed or masked (C1). Often, residues with low PP form stretches, and these might be joined by PREQUAL and this information is also printed (C2). By default, PREQUAL fully removes low PP residues in the N- and C-termini, whereas residues in the “core” region are masked with a “X” character (C3).

D Name of main output file.


```

-----
PREQUAL v.1.02 by Simon Whelan
-----
A There are 251 sequences of max length 746
  Prepping pairHMM ... done
  Collecting subset of posterior probabilities based on closest 10 sequences determined by Kmers
    This may take time for larger data sets: B1
B Creating collection sets of PPs based on Kmer distances
  \ 251 / 251 ... done B2
  Getting posterior probabilities:
    ... Done Get Posteriors
  Outputting posterior probabilities to example.fasta.filtered.PP... done B3
C Performing filtering:
  Applying standard threshold 0.994 resulting in 3494 residues removed C1
  Extending filtered regions with width of 10 ... 19 additional regions removed C2
  Applying front/back trimming for runs of 3 resulting in 90 sections removed C3
D Outputting results:
  Outputting filtered amino acid sequences to example.fasta.filtered

Computation complete

===== Summary =====
E
#Sequences      Original   Filtered   %Retained
#Residues      137880    134290    97.4%
F Analysis may have some problems. Warnings output to example.fasta.warning

Filtering complete!

```

Fig. 2 Typical screen output of PREQUAL, showing basic information of input dataset, analysis, and results. See main text for detailed description

E Summary of the analysis showing the number of sequences and residues in the input (“Original”) and output (“Filtered”). Note that individual sequences might be entirely removed, in which case a message will be printed to “.warning.”

F Sometimes, PREQUAL will generate an additional warning file. This information is important and should be inspected. Examples of possible warnings might be “WARNING: 86.94% of sequence removed/missing for [5] Seq1” or “WARNING: Fully removed sequence [5] Seq2.”

3.3 Using PREQUAL with DNA Sequences

When the input data are nucleotide sequences of protein-coding genes, these are first translated *in silico* upon automatic selection of the appropriate genetic code (this can also be provided by the user), sequences are then filtered at the amino acid level as described before, and the corresponding filtered DNA sequences are printed to the output.

A simple run in PREQUAL with DNA sequences might look like this:

```
./prequal my_seq_DNA.fasta
```

The program will first confirm that the input consists of DNA sequences by writing to the screen: “Found only DNA sequences. Doing translations.”

In this case, the default consists of at least four files:

`my_seq_DNA.fasta.dna.filtered`: main output containing the masked nucleotide sequences.

`my_seq_DNA.fasta.filtered`: corresponding masked amino acid sequences.

`my_seq_DNA.fasta.translation`: information regarding the translation. From nucleotide to amino acid sequences and the applied genetic code.

`my_seq_DNA.fasta.PP`: internal file containing the calculated PP.

`my_seq_DNA.fasta.warning`: an additional warning file will most likely be generated, containing important information about the sequences.

While using PREQUAL with nucleotide sequences of protein-coding genes is possible (and highly recommended), some restrictions apply. All sequences must be in the first reading frame and must have all its codons complete, and sequences must not contain stop codons (*). In case of any ambiguous nucleotide (not A, C, G, or T), the entire codon will be ignored and treated as gaps in the output. If PREQUAL detects that the input file contains DNA sequences, it will try to automatically select the right genetic code to translate the DNA sequences. This automatic code selection is heavily guided by stop codons, therefore the need to ensure that no internal stop codons are present in the input. Small differences in the genetic code when doing the translations should not strongly affect the filtering step, but it is highly recommended that the user checks the automatically selected genetic code, specified in the translation file. It is also possible to enforce the use of the universal genetic code using the option “`-forceuniversal`” (see below). When using PREQUAL with DNA data, it is highly recommended to check the warning and translation files to ensure that the filtering has been performed adequately.

3.4 Advanced Usage

In the following, we describe all available settings for fine-tuning the behavior of PREQUAL and adapt it to the user’s preferences. A short description of all available options can be obtained in the command line by typing “`--h all`.” (see also **Notes 1** and **2**).

3.4.1 Options Affecting the Definition of Core Regions and Filtering

To account for the different evolutionary patterns observed across most proteins, PREQUAL defines core and flanking regions. A core region is defined as the central part of the protein that is evolutionarily relatively well conserved, and its flanking regions,

attributable to N- and C-termini, are allowed to be less conserved. In practice, core regions are defined by the presence of at least three contiguous residues with high PP. By default, residues with low PP within the core region are masked with an “X” whereas those in the flanking (noncore) regions are simply removed.

The default behavior regarding the definition of core regions and filtering can be controlled with the following options:

- corerun** *X*: Defines the number (*X*) of contiguous residues with high PP that are used to define a core region. Low values of *X* will make the program more generous at defining the core, whereas high values will make the program more conservative. Default is 3.
- nocore**: No core region is defined. Default is to define a core region, which is defined by the presence of at least three consecutive high-PP residues.
- removeall**: Removes all low PP residues rather than only those in the flanking regions. This option should be used with caution. The masking of residues within the core region maintains the original distance between residues, thereby facilitating the inference of the original positional homologies. The complete removal of residues can negatively affect MSA.
- corefilter** *X*: Defines the character used to mask residues in the core region. Default is “X” and alternatives can only be a single character. Alternative characters might help in the visualization.
- noremove****repeat**: By default, PREQUAL will attempt to remove long identical repeats that can occur within individual sequences, most often due to sequencing or assembly errors. By selecting this option, repeats will not be removed (*see* also **Note 3**).

3.4.2 Options Related to DNA Sequences

The following options affect how PREQUAL deals with protein-coding DNA sequences:

- nodna**: By default, PREQUAL will try to guess whether the input data are amino acids or nucleotides. Using this option forces PREQUAL to read input as amino acid sequences. This might be useful for proteins extremely enriched in alanine (A), cysteine (C), glycine (G), and/ or threonine (T).
- forceuniversal**: This option forces PREQUAL to always use the universal code. By default PREQUAL will attempt to choose the right genetic code for translating the DNA sequences by detecting codons that differ from the standard code.

3.4.3 Options Affecting Output Formats

The output files and naming conventions can be modified using the following settings:

- outsuffix X: By default PREQUAL prints out filtered data to the file “.filtered” using the input file as a base name. This option will change the suffix of the file containing the filtered sequences.
- dosummary: This will generate a new output file “.summary” containing a summary of the analyses, including information on the proportion of residues removed per sequence (in total and within core regions) and some helpful cutoffs of the proportion of removed data with their associated PP thresholds, which can be informative to select a custom cutoff.
- dodetail: This will generate a new output file “.detail” where, for each sequence and residue, the following information will be printed out: estimated PP (maxPP; range 0,1), whether the residue has been filtered out (toRemove; 0 = FALSE; 1 = TRUE), and whether it is within the core region (Inside; 0 = FALSE; 1 = TRUE) (see Fig. 3). This file can be used to inspect the PP values of specific residues to personalize the filtering options.
- noPP: Do not output the posterior probability file (“.PP”). Note that this will require to recalculate PPs at every run of PREQUAL (see also Note 1).

3.4.4 Options Affecting Posterior Probabilities and Filtering

The following options might be used to modify the residue filtering as well as the heuristics used for the calculation of PPs:

- filterthresh X: Modifies the PP threshold. This is the main filter in PREQUAL, and any residue with a PP below this

#	[seq_pos]	seq_character	maxPP	ToRemove	Inside
>Seq1					
[0]	M	1.0000 0	1		
[1]	S	1.0000 0	1		
[2]	D	1.0000 0	1		
[3]	K	1.0000 0	1		
[4]	L	1.0000 0	1		
[5]	T	1.0000 0	1		
[6]	R	1.0000 0	1		
[7]	I	1.0000 0	1		
[8]	A	1.0000 0	1		
[9]	I	1.0000 0	1		
[10]	V	1.0000 0	1		

Fig. 3 Example of a detailed output file generated with “-dodetail” option. See main text for detailed description

threshold will be filtered out. The default value ($X = 0.994$) has been defined based on analyses of simulated data and validated on a wide range of real sequence datasets (*see* Subheading 4). In principle, `filterthresh` can take any value between 0 and 1, corresponding to situations where all residues would be kept or filtered out, respectively.

- `filterprop X`: An alternative way of filtering the data is by defining the proportion of the original data ($X\%$) that the user is willing to loose. When using this option, PREQUAL will automatically adjust the PP threshold accordingly. In practice, PREQUAL will often filter out a slightly higher proportion of data than specified because of the way regions of low confidence are joined together and how N- and C-termini are dealt with. This option will also print out to screen several cutoffs of the proportion of removed data with their associated PP thresholds, which might be informative to select the desired cutoff. The options “`-filterprop`” and “`-filterthresh`” may not be used together.
- `pptype closest/longest/all [Y]`: This setting modifies the algorithm that chooses the subset of sequences that will be used to calculate PPs for each individual sequence. The default is to compare the ten closest sequences defined by Kmer distances (“`-pptype closest`”). The number of closest relatives considered might be raised to improve the accuracy of the PPs by specifying Y (e.g., “`-pptype closest 20`”). It is also possible to specify that PPs are calculated with the ten longest sequences (“`-pptype longest`”; the number of longest sequences may also be changed with Y). We recommend to use the latter option with caution because the longest sequences might contain the most errors. A last possibility is to consider all sequences (“`-pptype all`”). The last option might be very slow, particularly for datasets with many sequences.
- `filterjoin X`: The default behavior of PREQUAL is to join low PP residues into masked stretches if these are separated by fewer than ten residues. This option allows modifying the number of residues required between two low PP residues (X) so that the entire sequence stretch is filtered out. Low values of X are more generous and will keep more data, whereas high values of X are more stringent and will filter out more data.
- `nofilterlist X`: This option allows to provide a file “ X ” containing a list of full sequence names (i.e., corresponding to the FASTA headers) to be ignored by PREQUAL during the masking step. This option might be useful if some sequences in the input file are expected to be highly divergent a priori, which might lead to incorrectly filtering out too much legitimate data (*see* **Note 4**).

`-nofilterword X`: Similar to the above option, it is possible to define a file “X” that contains a list of key words that might appear within sequence names (i.e., within the FASTA headers) and these will not be masked by PREQUAL.

4 Benchmarking

The accuracy and efficiency of PREQUAL in detecting true errors were tested using simulated error-free sequences, which were corrupted by including two types of errors at known—but random—positions, and tracking the number of errors (true and false) that were detected by PREQUAL [6]. Briefly, sets of 100 gene alignments were simulated in INDELible v.1.03 [14] using the WAG+Γ model ($\alpha = 0.5$ or 1.8) on a published tree containing both long and short terminal branches [15]. Simulated alignments were 500 amino acids long and contained a core region (450 amino acids) flanked by more gappy regions (25 amino acids each) aiming to mimic typical exon alignments. Three sets of 100 alignments were simulated, each differing on the level of gappyness: low (0.01 gap rate for the core and 0.02 for flanking regions), medium (0.02 and 0.05), and high (0.1 and 0.2). Individual alignment files were then corrupted by randomly inserting errors in sequences (proportional to sequence length) and locations. The errors corresponded to random amino acids drawn proportionally from the residue frequencies of the WAG model. Errors were either inserted at random positions, mimicking misannotation errors derived from wrong gene models, or replaced parts of the original sequences, mimicking frameshifts produced by indels at the nucleotide level. Three error rates were tested: low (0.001 errors per amino acid, ~22 errors per file), medium (0.002: ~44 errors), and high (0.003: ~66 errors). In addition, different lengths of individual errors were also tested: 10, 20, and 30 amino acids for their final expected lengths (the actual lengths were drawn from a geometric distribution). Overall, a total of 108 experimental conditions were simulated including three sets of alignments of varying gappyness, three error rates, three error lengths, and two types of errors (misannotations and frameshifts). Simulated gene files were analyzed with PREQUAL v.1.0 and the results are summarized in Table 1.

Overall, PREQUAL performed well under all tested conditions but was particularly efficient at detecting errors inserted at random positions mimicking misannotations (>98% were captured). Accuracy (proportion of correctly identified true and erroneous residues) was in most cases >90%, although it lowered up to 73% when very gappy alignments were considered. Regarding frameshift errors that replaced parts of the original sequences, PREQUAL generally detected >93% of erroneous residues, except for short

Table 1
Performance of PREQUAL on simulated data

		Misannotations			Frameshifts		
		No. of characters	Accuracy (%)	Errors captured (%)	No. of characters	Accuracy (%)	Errors captured (%)
Gappyness	Low	2.339.354	95.47	99.67	2.250.400	95.60	93.21
	Mid	2.345.437	91.84	99.68	2.255.919	91.89	94.60
	High	2.362.243	73.10	99.76	2.272.380	73.14	96.61
Number of errors	Low	2.300.164	92.26	99.81	2.255.919	92.36	94.24
	Mid	2.390.366	91.47	99.78	2.255.919	91.49	94.62
	High	2.390.366	91.47	99.78	2.255.919	91.49	94.62
Expected error length	10 AA	2.296.403	91.77	99.16	2.255.919	91.90	68.97
	20 AA	2.390.366	91.47	99.78	2.255.919	91.49	94.62
	30 AA	2.389.967	91.99	99.87	2.255.919	91.98	98.15

Fig. 4 ROC curve showing the relationship between true-positive rates (TPR) and false-positive rates (FPR) for frameshift and misannotation errors based on simulated data. Dots on curves mark the critical point where $\geq 95\%$ of correct amino acids are retained and $>90\%$ of the errors are removed, corresponding to the default PP threshold of 0.994

error stretches of ~ 10 residues where 69% of errors were captured. The accuracy of frameshift detection followed the same trend as for misannotations. The proportion of legitimate residues removed by PREQUAL on error-free sequences was $\sim 7\%$ for most tested conditions.

Using a ROC curve, we derived the default PP threshold of 0.994 so that $\geq 95\%$ of correct amino acids were retained while removing $>90\%$ of frameshift and misannotation errors (*see* Fig. 4). The area under the curve (AUC), representing the performance of PREQUAL as classifier, was 0.965 and 0.992 for frameshifts and

misannotations, respectively. The default threshold derived from simulated data was further evaluated on various real phylogenomic datasets [3, 16, 17]. A careful inspection of the results suggested that the overwhelming majority of errors identifiable by eye were removed using the default PP threshold, showing its appropriateness for a wide range of datasets characterized by very different levels of sequence divergence, from ca. 20 to 2000 million years ago.

PREQUAL can handle typical datasets on a common laptop computer. For example, the analysis of 107 sequences with a maximal length of 675 amino acids took 42 s on a computer equipped with a 2.7 Ghz i7 processor, whereas a larger set of 272 sequences with a maximal length of 1149 amino acids took 139 s (*see also Note 5*).

5 Notes

1. Although the default settings should work well for most applications, it is possible to fine-tune PREQUAL's behavior to specific datasets and user needs. To facilitate this process, PP values (the computationally most expensive part) are stored into an output file (".PP") by default. When a PP file with the same root name as the input file is present in the same directory, PREQUAL will get the PP values from that file and only perform the filtering step, which is almost immediate. This functionality allows to experiment with different filtering options without additional computational cost.
2. PREQUAL is not designed to detect highly divergent paralogs. If a single divergent paralog exists per input file, PREQUAL might be able to filter it out if the evidence for homology (PP) is low. However, this will be almost impossible if two or more paralogs are present because the evidence of homology between them might be high and thus paralogs will be retained. Similarly, PREQUAL will have problems to remove primary sequence errors if these happen at similar relative locations in two or more sequences. Even though most errors are expected to not have this distribution, it can happen if, e.g., sequences contain primer sequences or they mistakenly incorporate the same intron due to a propagated annotation error. These situations violate the underlying assumptions of PREQUAL. In the presence of multiple divergent paralogs or errors at similar locations, PREQUAL will assume that the observed high similarity can only arise due to shared ancestry and therefore the affected residues will most likely be kept. Therefore, it is recommended to check the filtered data and to deal with paralogs with more appropriate methods.

3. The use of PREQUAL on repetitive sequences is not recommended. PREQUAL was designed to work with simple protein-coding homologous sequences. If input sequences contain homologous regions within the sequence itself, such as repeated domains or other tandem repeats, establishing the homology between the repeats becomes challenging. If dealing with such cases, it is recommended to carefully check the results and decide whether PREQUAL adequately dealt with the data at hand. Sometimes, repeated regions might occur within individual sequences by error (e.g., erroneous gene models). By default, PREQUAL will assume that long identical repeats are due to errors and it will try to remove them (printing a note to the warning file). This can be avoided using the “`-noremove-repeat`” option.
4. PREQUAL will have trouble to handle highly divergent sequences in a set of otherwise conserved proteins. This can be the case for sequences from parasites or prokaryotic homologs analyzed together with eukaryote sequences. In these cases, PREQUAL will likely mask many error-free—but divergent—residues. In order to overcome this problem, it is possible to provide a list of taxa that should be ignored during homology inference (*see* Subheading 3.4.4).
5. PREQUAL is computationally light and runs on a single CPU. With genome-scale data, it is routine to analyze hundreds or thousands of input files. Depending on the number and size of the input files, this might be done with a “for” loop in Bash. It is also possible to take advantage of parallelization on multicore systems, e.g., using GNU parallel [18] or incorporate PREQUAL into pipelines or workflow managers such as Snakemake [19].

Acknowledgments

We would like to thank Kazutaka Katoh for the possibility of contributing this chapter. Max E. Schön provided comments on an earlier version. II acknowledges the support from a Juan de la Cierva-Incorporación postdoctoral fellowship (IJCI-2016-29566) from the Spanish Ministry of Science and Competitiveness (MINECO). This work in the lab of FB is supported by a fellowship from Science for Life Laboratory. SW thanks the Carl Tryggers Stiftelse and Uppsala University for support.

References

1. Chatzou M, Floden EW, Di Tommaso P, Gascuel O, Notredame C (2018) Generalized bootstrap supports for phylogenetic analyses of protein sequences incorporating alignment uncertainty. *Syst Biol* 67(6):997–1009
2. Philippe H, Brinkmann H, Lavrov DV, Littlewood DTJ, Manuel M, Wörheide G et al (2011) Resolving difficult phylogenetic questions: why more sequences are not enough. *PLoS Biol* 9(3):e1000602
3. Irisarri I, Meyer A (2016) The identification of the closest living relative(s) of tetrapods: phylogenomic lessons for resolving short ancient internodes. *Syst Biol* 65(6):1057–1075
4. Schneider A, Souvorov A, Sabath N, Landan G, Gonnet GH, Graur D (2009) Estimates of positive Darwinian selection are inflated by errors in sequencing, annotation, and alignment. *Genome Biol Evol* 1:114–118
5. Di Franco A, Poujol R, Baurain D, Philippe H (2019) Evaluating the usefulness of alignment filtering methods to reduce the impact of errors on evolutionary inferences. *BMC Evol Biol* 19(1):21
6. Whelan S, Irisarri I, Burki F (2018) PREQUAL: detecting non-homologous characters in sets of unaligned homologous sequences. *Bioinformatics* 34(22):3929–3930
7. Criscuolo A, Gribaldo S (2010) BMGE (Block Mapping and Gathering with Entropy): a new software for selection of phylogenetic informative regions from multiple sequence alignments. *BMC Evol Biol* 10(1):210
8. Capella-Gutiérrez S, Silla-Martínez JM, Gabaldón T (2009) trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics* 25(15):1972–1973
9. Castresana J (2000) Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Mol Biol Evol* 17(4):540–552
10. Ali RH, Bogusz M, Whelan S (2019) Identifying clusters of high confidence homologies in multiple sequence alignments. *Mol Biol Evol* 36(10):2340–2351
11. Durbin R, Eddy SR, Krogh A, Mitchison GJ (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
12. Wu M, Chatterji S, Eisen JA (2012) Accounting for alignment uncertainty in phylogenomics. *PLoS One* 7(1):e30288
13. Bogusz M, Whelan S (2017) Phylogenetic tree estimation with and without alignment: new distance methods and benchmarking. *Syst Biol* 66(2):218–231
14. Fletcher W, Yang Z (2009) INDELible: a flexible simulator of biological sequence evolution. *Mol Biol Evol* 26(8):1879–1888
15. Whelan NV, Kocot KM, Moroz TP, Mukherjee K, Williams P, Paulay G et al (2017) Ctenophore relationships and their placement as the sister group to all other animals. *Nat Ecol Evol* 1(11):1737–1746
16. MacLeod A, Irisarri I, Vences M, Steinfartz S (2015) The complete mitochondrial genomes of the Galápagos iguanas, *Amblyrhynchus cristatus* and *Conolophus subcristatus*. *Mitochondr DNA Part A* 27(5):3699–3700
17. Burki F, Kaplan M, Tikhonenkov DV, Zlatogursky V, Minh BQ, Radaykina LV et al (2016) Untangling the early diversification of eukaryotes: a phylogenomic study of the evolutionary origins of Centrohelida, Haptophyta and Cryptista. *Proc R Soc B-Biol Sci* 283(1823):20152802
18. Tange O (2015) GNU Parallel 20150322 ('Hellwig'). *USENIX Magazine* 36:42–47
19. Köster J, Rahmann S (2012) Snakemake: a scalable bioinformatics workflow engine. *Bioinformatics* 28(19):2520–2522
20. Larsson A (2014) AliView: a fast and lightweight alignment viewer and editor for large datasets. *Bioinformatics* 30(22):3276–3278