Imputor User Guide

Matthew Jobin

1. Overview

IMPUTOR is a software program written in the Python language for the purpose of identifying miscalled bases in sequence or variant data. Advances in next-generation sequencing have provided researchers with a wealth of data, but the data generated has proven variable in its fidelity to the original sequence (Bobo et al. 2016; Wall et al. 2014). Numerous software pipelines have been constructed in order to process data, with one important step being the assessment and correction of chemical or mechanical errors introduced by the NGS process. In some cases, these methods have used reference sequences derived from sequencing methods with greater coverage or reliability, while some have also implemented methods to impute missing data via the principle of parsimony (Wang et al. 2012; Poznik et al. 2013). IMPUTOR imputes missing or suspected mis-called bases via parsimony, identifying sites where a combination of mechanical error and conservative base calls may have misidentified the site.

2. Setup and installation

2.1. Python packages

IMPUTOR is written in Python 2.7 and requires some standard Python packages to run properly. Python distributions vary in what comes installed—in the case where a needed package has not been installed, the recommended method for installing packages is the PIP installer program. Following the given link should allow the user to install all needed Python packages. The list of needed packages for this release of IMPUTOR follows:

- biopython
- progressbar2

2.2. External programs

Two external maximum likelihood software programs can be used with IMPUTOR in order to construct phylogenetic trees. Please note that for these programs to work with IMPUTOR, they must be compiled and the executable placed in the same folder as IMPUTOR or elsewhere in the system's search path.

2.2.1. PhyML

Unlike methods of maximum parsimony, maximum likelihood methods make use of the branch lengths of a proposed tree .

2.2.1.1. How to install PhyML for use with Imputor

1. Download and install PhyML for your system as per the PhyML user guide.
2. Rename the resulting executable ‘phyml’ (no quotes).
3. Move your phyml executable to a location that is part of your system’s search path. On UNIX-like systems including macOS and Linux, you will usually want to add the line **export PATH="/your path/:$PATH**" to the .bashrc or the .bash profile located in your home directory (where your path is the path to the directory that contains PhyML binary).

2.2.2. RAxML

1. Download and install RAxML for your system as described in the RAxML user guide.
2. Compile the version best suited to your needs. Please see the RaXML docs about parallel computing, makefiles, etc.
3. Rename your compiled executable to 'raxmlHPC' (no quotes).
4. Move your raxmlHPC executable to a location that is part of your system's search path. On UNIX-like systems including macOS and Linux, you will usually want to add the line **export PATH="/your path/:\$PATH**" to the .bashrc or the .bash profile located in your home directory (where your path is the path to the directory that contains RAxML binary).

3. Input

IMPUTOR accepts input from two file types: FASTA files and VCF files. A reference sequence must also be input VCF format files, in order to generate the sequences for every sample. For VCF files, the sequence of the reference file is used to generate sequence from the defined variants inside the software, allowing either FASTA or VCF format output.

A phylogenetic tree is also necessary for the phylogeny-aware imputation performed by the software, and thus it is necessary either to import or generate such a tree. IMPUTOR provides four options for input: phyloxml import, tree construction by parsimony using Biopython, maximum likelihood via PhyML or maximum likelihood via RAxML. Software options necessary to modify the output of the tree have been included in IMPUTOR, however for detailed descriptions of their functions the individual software packages' under manuals will be the user's best guide (see below).

NOTE: Please MAKE SURE your input file has Unix-style line feeds for newlines!

3.1. Input options

Imputor handles the following input arguments:

- -file (required): The raw input data, in FASTA or VCF format. Note that VCF format files must include the optional FORMAT column with the GT subheader, and include columns where individual samples are marked.
- -tree: The method for importing or constructing a tree, with the following options:
    - *.xml: Input of a file with extension .xml will invoke reading as a phyloxml format tree (Han & Zmasek 2009).
    - PhyML: Invokes the PhyML software (Guindon et al. 2010).
    - RAxML: Invokes the RAxML software (Stamatakis 2014).
    - Pars: Invokes the RAxML software using the -y switch to return a maximum parsimony tree.
- -alpha: The value of the gamma shape parameter used in PhyML. Default is *e*.
- -rmodel: The model type for RAxML. Please see the RAxML user guide for instructions. Default is GTRCAT.
- -mutrate: The point mutation rate. Default is 8.7e-10.
- -nobackmutchk: Skips a check to restrict imputation of non-missing sites to those that are an apparent back-mutation.
- -msize: The number of neighboring sites that must be identical in order to impute their state to the target missing site. Default is 2.
- -nsize: The number of neighboring sites that must be identical in order to impute their state to the target non-missing site. Default is 3.
- -starttree: A starting tree for tree construction in RAxML.
- -genoqual: Impute if the Phred-scaled genotype quality (“GQ”) score of a site, encoded in VCF files, is below this value. Default is 30 (Lazaridis et al. 2014).
- -adthresh: Impute if the ratio of Allelic Depth scores (“AD”) for the site, with the numerator the accepted allele and the denominator the sum of the other

values, is below this value. Default is 0.66.

- -maxthreads: The maximum number of threads allowed for a Pthreads version of RAxML.
- -maxheight: The maximum number of steps up a tree allowable when searching for neighbors. Default is 2.
- -maxdepth: After ascending the tree to an ancestor of the target, the maximum number of steps downward that may be taken when searching for neighbors. Default is 2.
- -mincoverage: Minimum coverage to impute. This is useful if the user wants to screen out very low coverage sites.
- -passes: Number of times to check through data for possible imputations. Default is 1.
- -seqonly: Switch to simply print out the input sequence in FASTA format and exit.
- -exlocal: add ./ to all invocations of external programs to make sure it is invoked in the current working directory.
- -ncollect: Choose methods of neighbors collection (mono, distance, hops, rootward).
- -boot: Number of bootstrap replicates for RAxML. Switches RAxML algorithm type to rapid Bootstrap analysis or “-f a”.
- -runs: Number of separate runs in RAxML. Keeps RAxML algorithm type as new rapid hillclimbing or “-f d”.
- -threshold: Proportion of runs or bootstrap replicates which must impute to impute final sequence. Default is 0.95.
- -maxhops: Maximum number of “hops” or branches of a tree to traverse searching for neighbors if -ncollect set to *hops*. Default is 5.
- -maxn: Maximum number of neighbors that will be searched for monophyly if -ncollect set to *mono*. Default is 5.

- -maxjump: Maximum increase in branch distance (over previous branch traversed) allowed before terminating neighbor search if -ncollect set to *distance* or *hops*. Default is 5.

4. Phylogenetic tree construction

Imputation in IMPUTOR is phylogenetically aware, meaning that instead of imputing using a trusted reference sequence, imputation is done based on the principle of parsimony given a phylogenetic tree. Thus, the construction of the phylogenetic tree

5. Neighbor collection

IMPUTOR offers multiple methods for collecting the nearest neighbors used to determine whether a site should be imputed. The three methods—**rootward**, **hops**, and **distance**—each exist on a continuum of trade-offs for speed and accuracy (see below). Each methods can be invoked using a number of command-line arguments, each explained in its own subsection.

5.1. Rootward

The **rootward** method starts at the target site and then ascends toward the root of the tree. At each step, the method will then descend leaf ward to a maximum number of steps defined by **maxdepth**, collecting neighbors from all leaf nodes descending from that node until it reaches the threshold number (which can be different for a missing versus non-missing target site) or until it runs out of branches to search. If the threshold has not been reached, the method will ascend another step and descend again, until the threshold is reached or the method had ascended **maxheight** steps rootward.

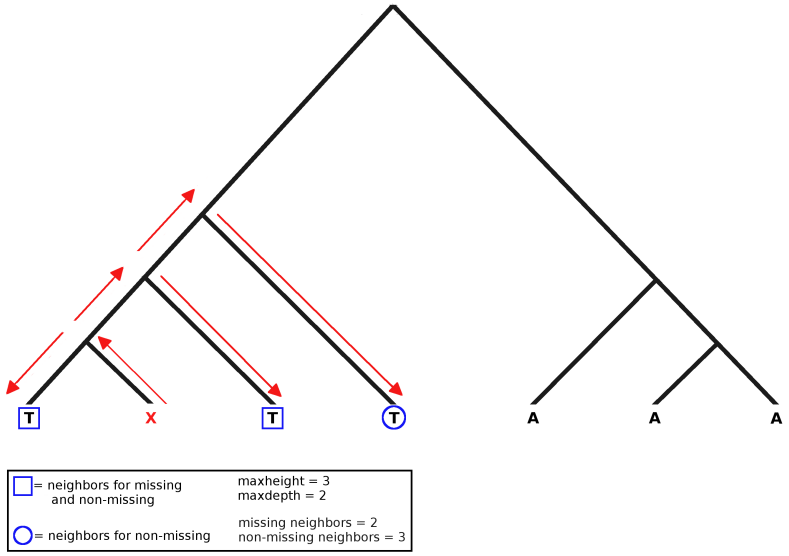

Figure 1: *Case 1, Rootward method*

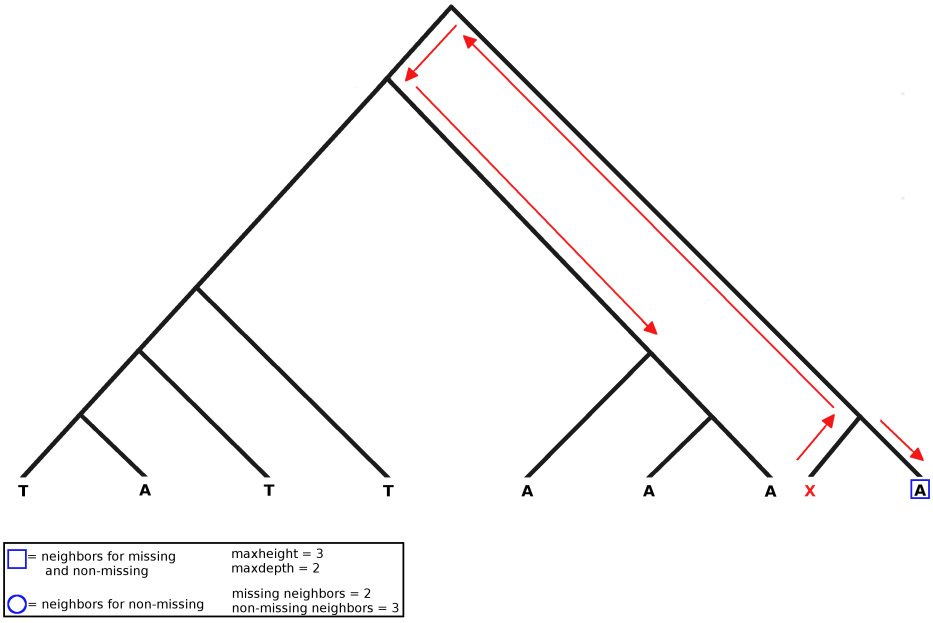

Figure 2: *Case 2, Rootward method*

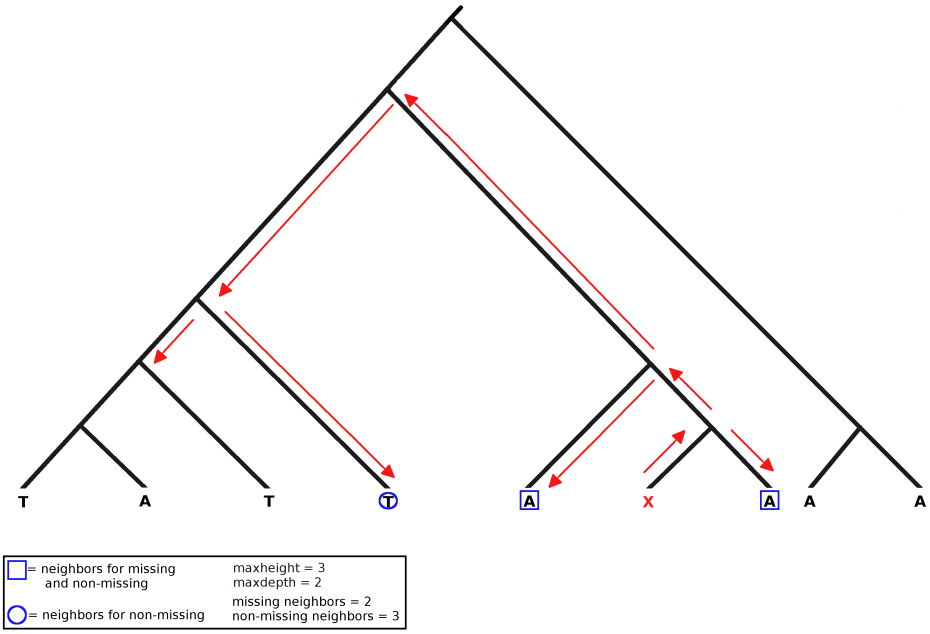

Figure 3: *Case 3, Rootward method*

5.2. Hops

The **hops** method searches from the target site rootward and leafward for neighbors, collecting the nearest neighbors determined by the total number of steps or hops from node to node along the tree. Apart from the maximum needed number of neighbors, two other parameters guide this methods. **Maxhops** determines the maximum number of hops that the software will take before ceasing a search for neighbors. **Maxjump** will cease the search for new neighbors if the branch length traversed in the most recent hop exceeds the length traversed in the previous hops by a multiple of the **maxjump** threshold. This is useful in decreasing the chances of finding neighbors from an isolated target site on the tree amongst groups that are not very close by to that target.

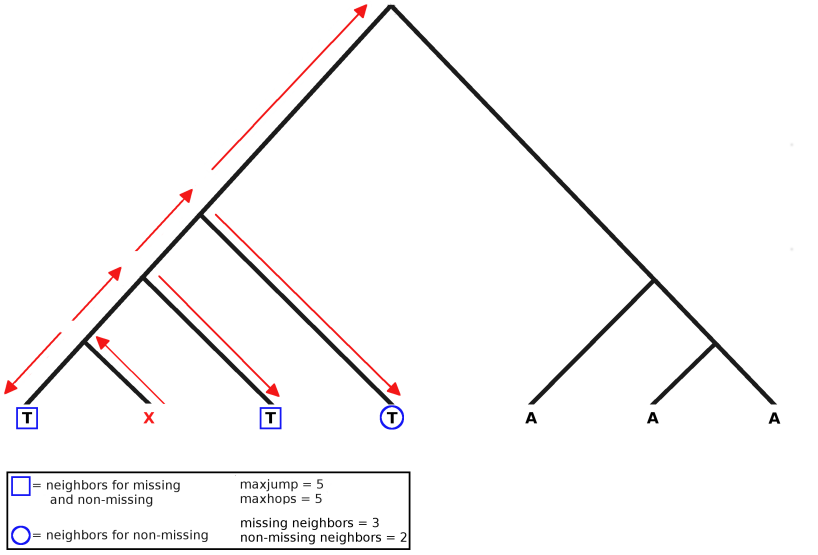

Figure 4: *Case 1, Hops method*

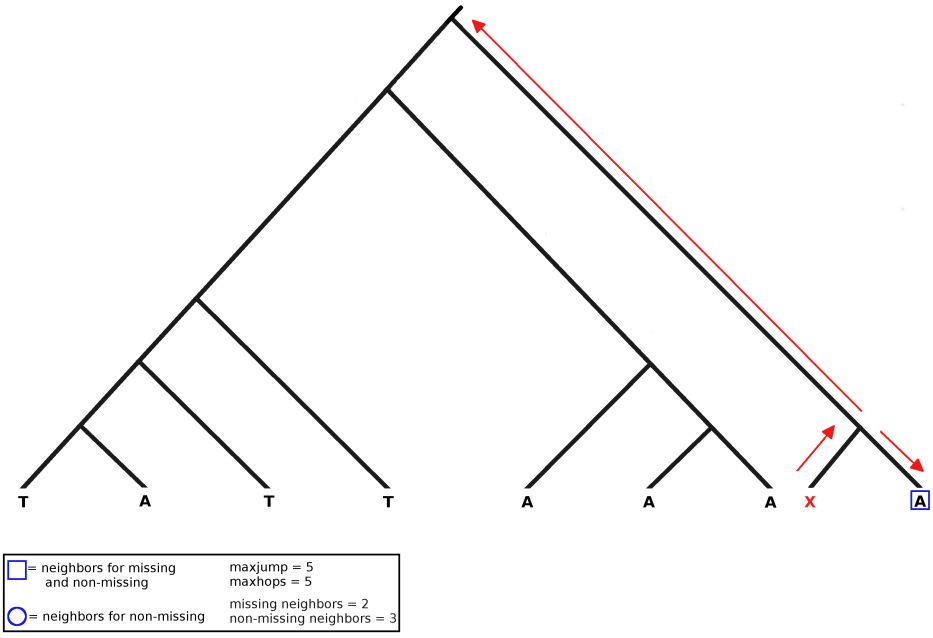

Figure 5: *Case 2, Hops method*

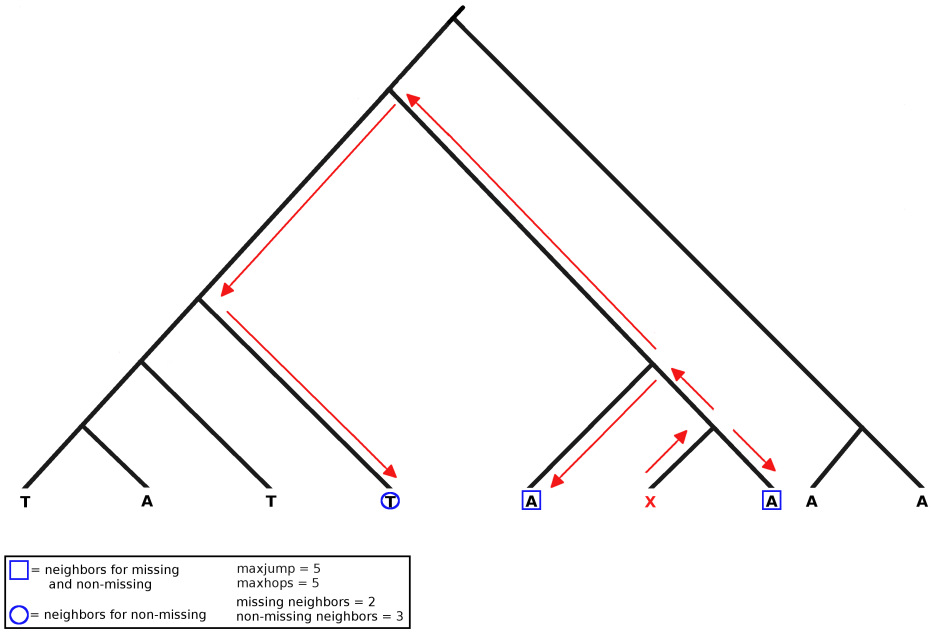

Figure 6: *Case 3, Hops method*

5.3. Distance

The **distance** method ordered the tree by branch length distance from the target, and selects the closet neighbors up to the stated threshold. To avoid the problem of isolated targets described in the previous section, the neighbor search wil terminate if the branch length searched in the current hop exceeds a multiple over the previous hop by a factor of **maxjump**.

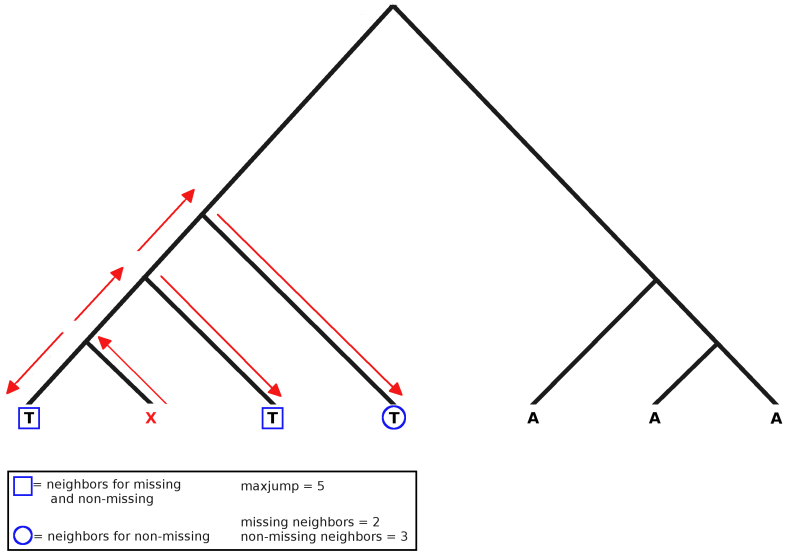

Figure 7: *Case 1, Distance method*

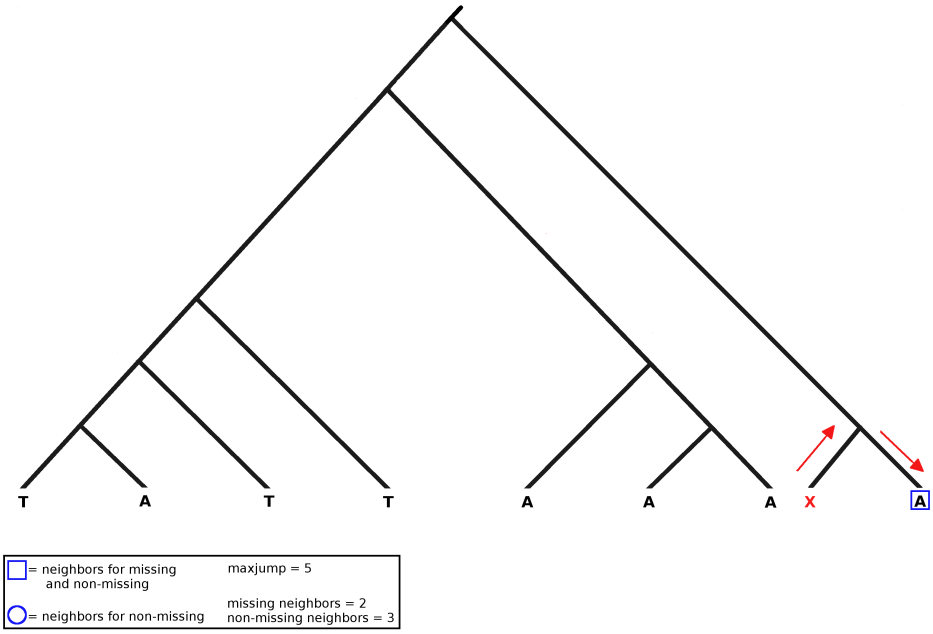

Figure 8: *Case 2, Distance method*

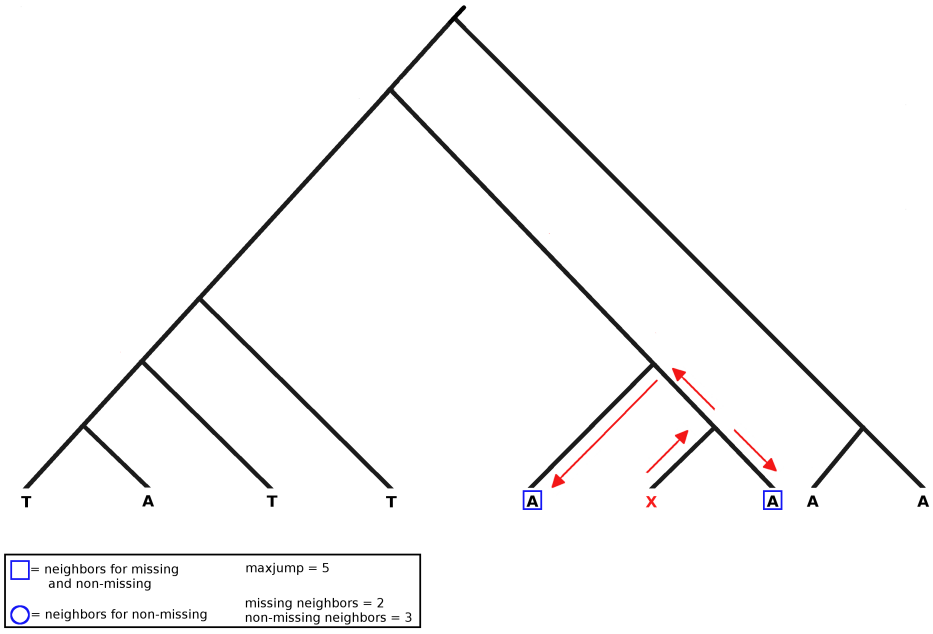

Figure 9: *Case 3, Distance method*

5.4. Comparison of methods

5.4.1. Speed

Speed of computation can become an important issue in choosing a method if the dataset is large or time is pressing. The following speed test was performed using a MacBook Pro with a 2.9 GHz Core i5 CPU:

Method	**Time (seconds)**
Rootward	0.133
Hops	4.74
Distance	13.6

Table 1: *Time taken to find neighbors 10000 times for a single target site*

5.4.2. Accuracy

Accuracy is, of course, far more important than speed… unless you're in a hurry. For our purposes, accuracy will be defined as how close an imputed sequence is to the true sequence, if you know what that true sequence is. In many real-world applications, of course, you will not know, which is why you are using IMPUTOR. Thus it helps to see what IMPUTOR can do when the true or original sequence is known or at least can be estimated to a fairly close approximation, in order to demonstrate the effectiveness of the software and its various methods. Below are two such types of self-tests, one with real data from father-son pairs and the other with simulated data which has been randomly altered to create an input file.

5.4.2.1. Father-son pairs on the Y chromosome

The Y chromosomal mutation rate has been calculated at 8.71×10^{-10} mutations per site per generation for the non-recombining section of the chromosome. Thus, along its 58 million base pair length, it is easily possible to have no differences between a father and a son. Using zero differences as a rough guideline (while acknowledging that there will of course be cases where one or mutations does exist between a father and son), we ran IMPUTOR on real data from a known pedigree including reported father and son pairs. This run of IMPUTOR was performed with the **rootward** neighbor-collecting method, the RAxML maximum-likelihood software and the following parameters: **maxdepth 4, maxheight 2, passes 5**.

Pair	Raw distance	Imputed Distance	Ratio
1	0.116	0	0
2	0.166	0	0
3	0.082	0	0
4	0.109	0.007	0.064
5	0.041	0	0

Table 2:

Pairwise distances between father-son pairs before and after imputation via IMPUTOR.

The pairs in the above tables were drawn from samples processed by next-generation sequencing which had a resultant number of missing sites as well as possible non-missing errors. The post-imputation profile is in line with expectations of father-son mutational profiles, with the majority showing zero differences but one pair showing a distance equivalent to a single mutation.

5.4.2.2. Simulated data

To test neighbor-collection methods for their ability to recover an original sequence, IMPUTOR was run using each method on altered versions of an original data file in FASTA format. This data was randomly generated by the forward similar SFS_CODE (Hernandez 2008), creating a file or 70 bases in length for 1000 haploid individuals. This file was then randomly mutated at with various constraints, including restricting the changes to missing, or non-missing, or allowing both to occur. These altered files were then run through IMPUTOR with the aforementioned neighbor-collection methods along with turning off and on checking for reversions. In the table below, Ratio means the ratio of similarity to the original file that the imputed file has to the randomly mutated file—thus, a ratio of 0.0 would mean all of the mutations were imputed accurately and the imputed file perfectly matched the original, undamaged file. The results were obtained by averaging over 10 different damaged files.

Type of Mutation	Method	Reversion check	Ratio
Missing	Rootward	Y	0.091
Non-missing	Rootward	Y	0.280
Both	Rootward	Y	0.206
Missing	Hops	Y	0.056
Non-missing	Hops	Y	0.251
Both	Hops	Y	0.199
Missing	Distance	Y	0.076
Non-missing	Distance	Y	0.212
Both	Distance	Y	0.167
Missing	Rootward	N	0.180
Non-missing	Rootward	N	0.294
Both	Rootward	N	0.204
Missing	Hops	N	0.147
Non-missing	Hops	N	0.290
Both	Hops	N	0.203
Missing	Distance	N	0.175
Non-missing	Distance	N	0.251
Both	Distance	N	0.170

Table 3:

Ratio of similarity of imputed file to randomly mutated file, relative to original file.

6. Imputation

A major challenge for Next Generation Sequencing is the separation of actual variation from errors that arise from the mechanical process of the method (DePristo et al. 2011).Imputation methods have been used to identify mistakenly called sites due to mechanical or laboratory error (Wang et al. 2012). These efforts are often made in order to “fill in” calls marked “missing” by the sequencing process (Lippold et al. 2014). One of the primary methods by which imputation is carried out for Next-Generation Sequencing (NGS) is by comparing NGS sequence reads to that of a set of reference sequences (Marchini & Howie 2010). An example of such a reference sequence set is Sanger sequencing, which is slower than NGS but has a lower error rate (Wall et al. 2014).

IMPUTOR imputes mutations for an input set of data via comparison of variants amongst near neighbors, via the principle of parsimony, wherein neighboring samples on a phylogenetic tree that are identical by descent (IBD) for a derived allele are unlikely to experience a reversion to the ancestral allele amongst one of their members. Under the principle of parsimony, originally introduced as the “principle of minimum evolution”, the course taken in evolutionary history is most likely to match the course that requires the fewest changes (Edwards & Cavalli-Sforza 1964). The principle of parsimony or minimum evolution has been applied via a phylogenetic tree for the purposes of imputing missing data (Poznik et al. 2013). In addition performing to this function, IMPUTOR searches for likely candidates for falsely negatively called mutations, which can appear in NGS as reversions aka back mutations. Previous studies imputing missing mutations have avoided introducing the possibilities for reversions due to their rarity (Wei et al. 2013).

One possible scenario for a false negative call is an apparent back mutation on a the phylogenetic tree. On such a tree, a back mutation will appear, from the point of view of any site if, when following its descendants we encounter a different allele, and then, continuing on the same descending path, encounter the original allele again (Requeno & Colom 2016). The same logic holds in the reverse direction, that if while ascending toward the root of a phylogenetic tree, one finds a different allele than the target allele, after which, on continuing to ascend, one finds a reversion to the target allele. Back mutations are not impossible, but rare, since the occurrence of a mutation on a lineage must be compounded with that of another on a branch of that lineage that happens to revert to the ancestral allele.

According to the principle of parsimony, should a target site differ from its three nearest neighbors, and should those three neighbors possess the same allele, and furthermore that target site be found among more distant neighbors (implying a possible reversion), the more likely scenario requiring fewer mutations would be the one generating the putative reversion than the mutations necessary to separately generate the mutations in its non-matching near neighbors. The length of the private branch upon which the target sequence sits represents the proportion of changes in that sequence relative t the sequence size. That length, or distance, multiplied by the rate of mutation, is the likelihood that a reversion has truly occurred, since at a constant rate of mutation the branch implies time. A short private branch is less likely to have provided the opportunity for a true reversion to have occurred, and is thus more likely that an identified reversion is the result of NGS sequencing error or an error of some other form.

IMPUTOR is designed to err on the side of caution, in other words to avoid imputation when the set criteria are not strictly met. The decision rule for imputation involves each sequence and its closest neighbors on the constructed or imported phylogenetic tree. IMPUTOR searches site by site for each sample, iterating through the list of samples in random order and implementing the following decision rule:

1. For each target, iterated in random order:
    1. Step back toward the root of the tree collecting nearest siblings of the target
    2. Keep stepping back until root is reached or threshold number of neighbors found not non-missing (default = 3) and missing (default=2) sites.
2. For each variant in the set of all variants in the input file:
    1. If the -mnm switch is on, the target is missing and it has two neighbors, one of which is missing, remove the missing neighbor to allow imputation
    2. Else if the nearest neighbors do not match, do not impute.
    3. If there are no neighbors that fit the given criteria, do not impute.
    4. If the target is marked as missing:
        1. If there are less than the given threshold missing neighbor size, do not impute.
        2. If the neighbors are all missing, do not impute.
        3. Else impute to state of neighbors (Poznik et al. 2013).
    5. Else if the target is marked as non-missing:
        1. If the target has too few neighbors as defined by the user (see input options for minimum neighbors and tree search parameters), do not impute.
        2. If the neighbors are all marked as missing data, do not

impute.

3. If the target does not match its neighbors:
    1. If the coverage is equal to or less than the minimum threshold, impute to state of neighbors.
    2. If the allelic depth quotient is below the assigned threshold, impute to state of neighbors.
    3. If the genotype quality score is below the assigned threshold, impute to state of neighbors.
    4. If checks for reversion are deactivated by the user, impute to state of neighbors.
    5. Else:
        1. If the target state is not found farther away than the three nearest neighbors (i.e. a potential reversion) then do not impute the target (Requeno & Colom 2016).
        2. Else impute to state of neighbors.
4. Else, do not impute.

7. Output

IMPUTOR provides output information to the console for the researcher to examine during run-time. After the run completes, a number of possible files can be written to storage, depending on the options chosen from the list in the next section.

7.1. Output options

- -out: The type of output file requested. Options are: fasta or vcf. Defaults to fasta.
- -outimpute: Write the list of imputed mutations to <inputfile minus extension> + “impute.txt”. Defaults to true.
- -outtree: The output file format of the phylogenetic tree, with the following options:
    - newick: (default) Newick format
    - nexus: Nexus format
    - phyloxml: phyloxml format
- -impout: Output list of imputed sites and their likelihoods. Default is true.

7.2. Output files

Depending on the -verbose switch, some or all of the following files will be generated by IMPUTOR, with <filename> standing for the

- <filename>-seqout.fasta: The sequence with imputations, in FASTA format.
- <filename>-impout.txt: A tab-delimited text file of all imputations (and, if verbose switch is on, all failed imputations). The list includes the reason why each imputation was or was not made.
- <filename>-indata.fasta: A FASTA file of the data as input. Useful if the input file was in VCF format.
- <filename>-outtree.newick, .nexus or .phyloxml: Best tree found by tree-constructing software.
- <filename>-indata-ref.txt: Reference sequence, if given.
- <filename>-indivout.txt: List of how many and which imputations made for each individual.

8. References

Bobo, D. et al., 2016. *False Negatives Are a Significant Feature of Next Generation*

Sequencing Callsets, Cold Spring Harbor Labs Journals. Available at: http://biorxiv.org/lookup/doi/10.1101/066043.

DePristo, M.A. et al., 2011. A framework for variation discovery and genotyping using next-generation dNA sequencing data. *Nature Genetics*, 43(5), pp.491–498. Available at: http://www.nature.com/ng/journal/v43/n5/full/ng.806.html%3FWT.ec_id=NG-201105.

Edwards, A.W.F. & Cavalli-Sforza, L.L., 1964. Reconstruction of evolutionary trees. *Syst. Assoc. Publ.*, No. 6, pp.67–76.

Guindon, S. et al., 2010. New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phyML 3.0. *Syst. Biol.*, 59(3), pp.307–321. Available at: http://sysbio.oxfordjournals.org/content/59/3/307.full.

Han, M.V. & Zmasek, C.M., 2009. PhyloXML: XML for evolutionary biology and comparative genomics. *BMC bioinformatics*, 10(1), pp.356–6. Available at: http://www.biomedcentral.com/1471-2105/10/356.

Hernandez, R.D., 2008. A flexible forward simulator for populations subject to selection and demography. *Bioinformatics*, 24(23), pp.2786–2787. Available at: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btn522.

Lazaridis, I. et al., 2014. Ancient human genomes suggest three ancestral populations for present-day Europeans. *Nature*, 513(7518), pp.409–413. Available at: http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pubmed&id=25230663&retmode=ref&cmd=prlinks.

Lippold, S. et al., 2014. Human paternal and maternal demographic histories: Insights from high-resolution y chromosome and mtDNA sequences. *Investigative*

Genetics, 5(1), p.13. Available at: http://investigativegenetics.biomedcentral.com/articles/10.1186/2041-2223-5-13.

Marchini, J. & Howie, B., 2010. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics*, 11(7), pp.499–511. Available at: http://dx.doi.org/10.1038/nrg2796.

Poznik, G.D. et al., 2013. Sequencing Y chromosomes resolves discrepancy in time to common ancestor of males versus females. *Science*, 341(6145), pp.562–565. Available at: http://www.sciencemag.org/cgi/doi/10.1126/science.1237619.

Requeno, J.I. & Colom, J.M., 2016. Evaluation of properties over phylogenetic trees using stochastic logics. *BMC bioinformatics*, pp.1–14. Available at: http://dx.doi.org/10.1186/s12859-016-1077-7.

Stamatakis, A., 2014. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9), pp.1312–1313. Available at: http://bioinformatics.oxfordjournals.org/content/30/9/1312.full.

Wall, J.D. et al., 2014. Estimating genotype error rates from high-coverage next-generation sequence data. *Genome Research*, 24(11), pp.1734–1739. Available at: http://genome.cshlp.org/content/24/11/1734.full.

Wang, J.R. et al., 2012. Imputation of Single-Nucleotide Polymorphisms in Inbred Mice Using Local Phylogeny. *Genetics*, 190(2), pp.449–458. Available at: http://www.genetics.org/cgi/doi/10.1534/genetics.111.132381.

Wei, W. et al., 2013. A calibrated human y-chromosomal phylogeny based on resequencing. *Genome Research*, 23(2), pp.388–395. Available at: http://genome.cshlp.org/content/23/2/388.full.