AAF User Manual

Huan Fan, Anthony Ives, Yann Surget-Groba and Chuck Cannon

hfan22@wisc.edu

Feb 2016

Table of Contents

Introduction

AAF (alignment and assembly-free) is a free software package that reconstructs phylogeny from next-generation sequencing data without assembly and alignment. It takes raw sequencing reads from each sample altogether and generates a distance matrix based on the proportion of shared k-mers between each sample and reconstruct a phylogeny based on the distance matrix.

AAF is mainly designed for big Eukaryotes genomes. Therefore we divided the whole reconstruction process into two major steps: 1) *k*-mer counting and 2) distance calculation and phylogeny reconstruction. We have two separate python scripts taking care of the two steps respectively: 1) **aaf_phylokmer.py 2) aaf_distance.py.** There are 3 more optional scripts in the AAF package. One for trimming excessive tips of the phylgenye generated due to sequencing error and imcomplete coverage (aaf_tip.py); two for doing bootstraps for the phylogeny constructed (nonparametric_bootstrap.py and parametric_bootstrap.R). In the rest of the manual we will introduce their usage and options respectively.

We have included two tutorials in this manual. One is a dummy dataset with 10 species and short genomes and the other is a real dataset with 21 tropical tree genomes as described in the AAF paper. The first one is used to showcase how to organize data and run the first scripts. The second one is used to demonstrated possible issues while dealing with real and big dataset, with special focus on parameter selection including *k* and filtering. There is a separate section detailing the reasoning for optimal parameter selection as well.

For the most recent version of AAF, please visit https://sourceforge.net/projects/aaf-phylogeny/

Requirements

AAF can be used on a UNIX system (Linux, OsX...) with Python 2.7+ and higher versions (including Python 3.X+), and g++/gcc compilers. Biopython (http://biopython.org/wiki/Main_Page) is required for the non-parametric bootstrap, and R (http://cran.r-project.org/) and the R package 'ape' are required for the parametric bootstrap.

Installation

0. Decompress the zip file downloaded from http://sourceforge.net/projects/aaf-phylogeny with the most recent version.

1. Compile kmer_count(x) and kmer_merge as follows. "path_to_AAF" stands for your path to the AAF folder generated by decompressing AAF.tar.gz.
    a. path_to_AAF/AAF$ cd phylokmer
    b. path_to_AAF/AAF/phylokmer$ make
    c. Add kmer_count(x) and kmer_merge to your PATH or working directory
2. Compile fitch_kmerX, consense and treedist
    a. path_to_AAF/AAF$ cd phylip_src
    b. path_to_AAF/AAF/phylip_src$ make all

c. Add fitch_kmerX and consense to your PATH or working directory

Usage and options

(See tutorials below for examples):

1) aaf_phylokmer.py

Usage: aaf_phylokmer.py [options]

Options:

--version	show program's version number and exit
-h, --help	show this help message and exit
-k KLEN	k-mer length, default = 25
-t NTHREADS	number of threads to use, default = 1
-n FILTER	k-mer filtering threshold, default = 1
-o OUTFILE	output file, default = phylokmer.dat.gz
-d DATADIR	directory containing the data, default = data/
-G MEMSIZE	total memory limit (in GB), default = 4
-W withKmer	include k-mers in the shared k-mer table
-s	only print commands, do not run them

Detailed description of options:

-k KLEN: *k*-mer size. Larger *k* will decrease the probability of two identical *k*-mers from different parts of the genome (*k*-mer homoplasy) while increase the probability of *k*-mers containing sequencing errors and multiple evolutionary events such as substitutions or indels. See more details in the parameter selection section. Set at 25 by default.

-t NTHREADS: number of threads to use. Depends on how many cores are available on your machine. Set at 1 by default.

-n FILTER: how many times a *k*-mer needs to be in the sample to be counted as present. This serves as the filter for singletons, which could be the result of sequencing error. See more details about the parameter selection section. Set at 1 by default.

-o OUTFILE: output filename. If you would like your output file to be compressed, provide a name that ends with .gz. Otherwise it will not be compressed. The default output file is phylokmer.dat.gz, which is compressed.

-d DATADIR: directory containing the data. Users should strictly follow the data structure required by AAF. Sequence files for each sample need to be in one directory named after that sample. Therefore, there will be N directories for N samples and the name of the directories will be the names displayed in the final phylogenetic tree. All the sample folders should be placed into the same directory, which will be your data directory requested by aaf_phylokmer.py. Accepted extensions for sequence files include: .fa(sta)(.gz), .fq(.gz), and .fastq(.gz). See the "data" directory in the package as an example.

-G MEMSIZE: the total memory allowance. Each kmer_count thread has G/t memory allowance. Set at 4G by default.

-W WITHKMER: to include *k*-mers in the shared *k*-mer table. When the final goal is to construct a phylogeny, we do not need to know the specific patterns of each *k*-mer. Therefore by default in the shared *k*-mer table only the frequencies of *k*-mers are kept. However if there's downstream analysis of *k*-mers with a certain pattern, *k*-mers need to be kept. Use -W to keep the *k*-mers.

-s SIM: This will print out the commands that are going to run without executing them.

2) aaf_distance.py

Usage: aaf-distance.py [options] -i <input filename>

Options:

--version	show program's version number and exit
-h, --help	show this help message and exit
-i IPTF	input file, default = phylokmer.dat.gz
-t NTHREADS	number of threads to use, default = 1
-G MEMSIZE	max memory to use (in GB), default = 1
-o OTPF	prefix of the output files, default = aaf
-f COUNTF	k-mer diversity file, default = phylokmer.dat.wc

Detailed description of options:
-i IPTF: input file. The shared *k*-mer table generated from aaf_phylokmer.py. The default file is phylokmer.dat(.gz)

-t NTHREADS: number of threads to use. Depends on how many cores are available on your machine. Set at 1 by default.

-G MEMSIZE: the total memory allowance in GB. Set at 4G by default.

-o OTPF: prefix of the output files, including the distance matrix(.dist) and the phylogenetic tree(.tre). Default is set as "aaf".

-f COUNTF: wc file generated from kmer_count. This file contains the *k*-mer diversity of each sample. The default is phylokmer.dat.wc

3) aaf_tip.py

Usage: aaf_tip.py [options] -i <input tree file> -k <kmer size> --tip <information for tip correction>

Options:

--version	show program's version number and exit
-h, --help	show this help message and exit
-i IPTF	tree file to be trimmed
-k KLEN	k-mer size used for constructing the input tree
--tip=TIP_FILE	tip setting file, default = tip_file_test.txt
-n	k-mer filtering was on for tree construction
-f COUNTF	k-mer diversity file, default = phylokmer.dat.wc

Detailed description of options:

-i IPTF: input tree file. The tree file whose tips you would like to trim.

-k KLEN: the *k* that was used to construct the input tree.

--tip TIP_FILE: To trim the excess tips caused by incomplete coverage and sequencing errors requires additional info on the average coverage, read length and sequencing error of each sample. Put this information into a tab delimited text file in the format of tip_info_test.txt. See suggestions on estimation of coverage and sequencing error in Parameter Selection section.

-n: add it to the command if filter was used during the tree construction.

-f COUNTF: wc file generated from kmer_count. This file contains the *k*-mer diversity of each sample. The default is phylokmer.dat.wc

4) nonparametric_bootstrap.py

Usage: nonparametric_bootstrap.py [options]

Options:

-h, --help	show this help message and exit
-k KLEN	k-mer length, default = 25
-t NTHREADS	number of threads to use, default = 1
-n FILTER	k-mer filtering threshold, default = 1
-f SEQFORMAT	format of input files, FA|FQ, default = FA

-o OUTFILE	k-mer table name, default = phylokmer.dat.gz
-d DATADIR	directory containing the data, default = data/
-G MEMSIZE	total memory limit (in GB), default = 4
--S1=STAGE1	number of resampling of the reads, default = 0
--S2=STAGE2	number of resampling of each total kmer table, default = 0
-s	only print commands, do not run them

Detailed description of options:
-k KLEN: *k*-mer size. Set at 25 by default.

-t NTHREADS: number of threads to use. Depends on how many cores are available on your machine. Set at 1 by default.

-n FILTER: how many times a *k*-mer needs to be in the sample to be counted as present. Set at 1 by default.

-f SEQFORMAT: Format of the sequence files, FA or FQ. The default is set as FA.

-o OUTFILE: file name of the merged *k*-mer table. If you would like your *k*-mer table to be compressed, provide a name that ends with .gz. Otherwise it will not be compressed. The default output file is phylokmer.dat.gz, which is compressed.

-d DATADIR: directory containing the data.

-G MEMSIZE: the total memory allowance. Each kmer_count thread has G/t memory allowance. Set at 4G by default.

--S1: number of times to resample the reads for each sequence file. This is the first stage of our two-stage bootstrap. This bootstrap result shows the variance in sequencing error and incomplete coverage. Set at 0 by default, which means skip the first stage of bootstrap and only resample the *k*-mer table.

--S2: number of times to resample the total *k*-mer table generated from one instance of resampling of the reads. If --S1 is set to be 0, the resampling is on the real *k*-mer table generated from the original data. Set at 0 by default, which means skipping this step.

5) parametric_bootstrap.R

When it takes too long to bootstrap over large datasets, switch to the parametric bootstrap. This R script provides estimation of the variances in the two steps. It requires:
info file: containing read length, sequencing error and coverage and used in aaf_tip.py, default = tip_info_test.txt
nshare file: containing the number of shared kmers generated by aaf_distance.py (ends with _nshare.csv), default = test_nshare.csv

nreadboot: number of replicates, default = 10
k: *k*-mer length used in previous steps, default = 21
i.filter: filter threshold used, default =1.

Tutorial with dummy dataset

1) Decompress the pipeline and the test data (It will become available as soon as this paper is accepted from the AAF project page http://sourceforage.projects/AAF-phylogeny):
$tar xvfz AAF.tar.gz

2) Move to the phylokmer directory and compile kmer_count, kmer_countx, and kmer_merge
path_to_AAF/AAF$ cd phylokmer
path_to_AAF/AAF/phylokmer$ make
path_to_AAF/AAF/phylokmer$ cp kmer_count kmer_countx kmer_merge ../

3) Compile fitch_kmerX
path_to_AAF/AAF/phylokmer$ cd ../phylip_src
path_to_AAF/AAF/phylip_src$ make all
path_to_AAF/AAF/phylip_src$ cp fitch_kmerX consense ../
path_to_AAF/AAF/phylip_src$ cd ..

4) k-mer counting
path_to_AAF/AAF/$ python aaf_phylokmer.py -k 21 -d data -G 2

5) Constructing the phylogenetic tree
path_to_AAF/AAF/$ python aaf_distance.py -i phylokmer.dat.gz -o test -t 2 -G 2 -f phylokmer.dat.wc

6) Tip correction (optional)
path_to_AAF/AAF/$ python aaf_tip.py -i test.tre -k 21 --tip tip_info_test.txt -f phylokmer.dat.wc

7) Non-parametric bootstrap (optional)
path_to_AAF/AAF/$ python nonparametric_bootstrap.py -k 21 -t 2 -d data --S1 2 --S2 2

8) Parametric bootstrap (optional)
Set your working directory to the AAF folder and change the parameters in the “set parameters” section, including nboot, k, filter, info.file and n.table.file.
Within R console or terminal:
> source(“parametric_bootstrap.R”)

Description of output files

1) phylokmer.dat.gz (aaf_phylokmer.py): This output file will be inside the data folder. It starts with header information including *k*-mer size, filter frequency and sample list. After the header is a table with frequencies of a given *k*-mer from each sample on one line in alphabetical order of the sample names.
2) phylokmer.dat.wc (aaf_phylokmer.py): Inside the data folder. It contains the total *k*-mer diversity for each sample per line in alphabetical order of the sample names.
3) test_nshare.csv (aaf_distance.py): In the current working directory. It contains the number of shared *k*-mers for each pair of samples. This file is needed for the parametric bootstrap.

4) test.tre (aaf_distance.py): In the current working directory. It is the phylogeny you want!
5) test.dist (aaf_distance.py): In the current working directory. It is the distance matrix upon which fitch_kmerX infers the phylogeny (test.tre in this case).
6) tip_test.tre (aaf_tip.py): In the current working directory. It is the tree after tip correction.
7) consensus_trees_read_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains all the trees that were generated after each resampling of the reads.
8) consensus_trees_total_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains all the trees that were generated after both resampling of the reads and the *k*-mer table counted from those reads.
9) consensus_trees_table_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains all the trees that were generated after each resampling of the real *k*-mer table calculated from the original reads when the resampling of reads is skipped.
10) consensus_read_nonparametric.tre (nonparametric_bootstrap.py): In the current working directory. This is the consensus tree generated from consensus_trees_read_nonparametric by consense in PHYLIP.
11) consensus_total_nonparametric.tre (nonparametric_bootstrap.py): In the current working directory. This is the consensus tree generated from consensus_trees_total_nonparametric by consense in PHYLIP.
12) consensus_table_nonparametric.tre (nonparametric_bootstrap.py): In the current working directory. This is the consensus tree generated from consensus_trees_table_nonparametric by consense in PHYLIP.
13) consensus_outfile_read_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains the bootstrap ratio for the branches in consensus_read_nonparametric.tre
14) consensus_outfile_total_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains the bootstrap ratio for the branches in consensus_total_nonparametric.tre
15) consensus_outfile_table_nonparametric (nonparametric_bootstrap.py): In the current working directory. This file contains the bootstrap ratio for the branches in consensus_table_nonparametric.tre
16) consensus_*_*_parametric (parametric_boostrap.R): In the current working directory. See descriptions of their nonparametric counterparts.

Parameter Selection

Here we provide some guidelines in parameter selection using the dataset with 21 tropical trees as an example.

a. Optimal *k*

As we described in the manuscript, the selection of *k* is a trade-off between avoiding multiple mutations on one *k*-mer (which favors shorter k) and decreasing the chances of *k*-mer homoplasy (which favors longer k). For the primate dataset in the manuscript, we plot the theoretical predictions of the proportion of shared *k*-mers, *ph*, calculated from the observed frequency distribution of *k*-mers and the *ph* calculated without homoplasy (Fig. 2D) to help view the effect of different choices of *k*. This procedure led to the selection of *k* that corresponded to an accurate phylogeny. Therefore this figure serves as a good indicator for optimal *k*, and this choice can be further proved by constructing phylogeny with *k*-mer lengths larger than optimal, in order to check the phylogenetic consistency.

To plot the *ph* vs. *k* figure for your dataset, here is a checklist for the genome information that is needed:
i. Sample names

ii. Coverage
iii. Genome size
iv. GC content
v. *d* (genetic distance)
vi. Q_k

We are aware that this information might not be all available, and we provide coarse calculation methods for some of the categories.

ii. Coverage
There are multiple ways of estimating the sequencing coverage of your next-gen sequencing data. (1) If the genome size is known, coverage = total bp / genome size. (2) If the genome size is unknown, we can estimate the coverage by plotting the *k*-mer frequency distribution: "if a large fraction of *k*-mers occur c times, we can estimate the sequencing coverage to be approximately c and derive an estimate of the genome size from c and the total length of the reads." (Marcais and Kingsford 2011)). c will be the *k*-mer coverage. To get the base pair coverage, you need correct c using base_coverage = c * read_length / (read_length - *k*-mer_size + 1) (see https://groups.google.com/forum/#!topic/bgi-soap/xKS39Nz4SCE). (3) When the coverage is low or sequencing error rate is high, there will be no clear peak in the *k*-mer frequency distribution at c. This is actually the case for all the tropical tree species in our dataset except *Ficus vasculosa* (FV). A coarse estimation of the *k*-mer coverage will be the total number of k-mers (including multiple copies of the same *k*-mer) divided by k-mer diversity (number of *k*-mer that shows up at least once). Some assemblers (such as velvet, SOAPdenovo) report estimation of *k*-mer coverage as well.

Coverage information is also needed for tip correction.

iii. **Genome size**
You can try to check the genome size in databases such as Plant DNA C-values Database (http://data.kew.org/cvalues/) or Animal Genome Size Database (http://www.genomesize.com/). If it is not available for your species, you can do a rough estimate using total base pair divided by coverage.

iv. **GC content**
There are many tools to calculate the GC content of your samples. In the AAF package we have provided our own, gc.py in the utils folder. Biopython needs to be preinstalled.

v. ***d*** **(Genetic distance)**
The genetic distance of the group (average number of mutations per base pair) is used to set the scale of the vertical axis in the *ph* vs. *k* figure. Because the figure is used to find *k* on the horizontal axis, the conclusions from this figure about selecting *k* are mostly independent of the selection of d, so this selection does not need to be very accurate. A reasonable strategy is to guess d, or use the default 0.1, to select k. The subsequent phylogeny construction will give a good estimate of *d* from the distance matrix, which then could be used to plot the figure.
vi. $\boldsymbol{Q_k}$
There is more than one way of calculating the frequency distribution of *k*-mers. One of the easiest ways is to turn on the --stats option while counting the *k*-mers using jellyfish(Marcais and Kingsford 2011). However the maximum *k* that jellyfish can handle is 31. For k>31, use kmer_countx to count the *k*-mers, then calculate the frequency distribution of *k*-mers from the pkdat files (the output file of kmer_count($k \leq 25$) and kmer_countx(k>25)) using the pkdat2hist.py in the utils folder that is provided

in the tutorial folder.

After gathering all the information, we generated the *ph* vs. *k* figure for the 21 tropical trees dataset (Fig. S6) using the R code phVSk.R in the utils folder. The trend for all the red lines (estimated *ph* based on the Q_k for each species) stabilized for $k \geq 25$, and the difference between the red lines and the black dashed line continued to decrease with larger *k.* Therefore, we constructed phylogenies for *k* from 25 to 31, and because the phylogenies were identical for $k \geq 27$ (Fig.7), we selected 27 as the optimal *k* for the tropical trees dataset. The same phylogenetic topology was also obtained when *k*-mers were filtered to remove singletons. For *k* greater than 31, the topology within the *Ficus* group showed some small changes. We suspect that this is due to the loss of sensitivity to evolutionary changes when selecting *k*-mer lengths too long, especially for relatively small genomes (as the Ficus group has genome sizes less than half those of the other species).

To plot the *ph* vs. *k* figure for your dataset, simply replace the genome information for the tropical trees with the information for your own dataset in the beginning of the R script phVSk.R.

b. Filter or not

In deciding whether or not to filter *k*-mers (i.e., only including *k*-mers if they occur at least twice in a taxon), it is necessary to know the balance between loss of information through false *k*-mers that caused by sequencing errors if there is no filtering, and loss of information through removing true singletons if there is filtering (Fig. 5 in the manuscript). If there is a large range of coverages among taxa within a dataset, it is best to decide whether or not to filter based upon the taxon with the lowest coverage, because there is a large negative consequence of filtering low-coverage taxa (Fig. 5). For the tropical trees, we chose not to filter because more than half of the species have coverage less than 5.

c. Tip trimming (optional)

Information needed for tip trimming/correction includes: coverage, read length, and sequencing error. You should be able to get the performance of your sequencing platform from your sequencing company. For example Illumina claims that the error rate for Genome Analyzer is about 1% (http://res.illumina.com/documents/products/datasheets/datasheet_genomic_sequence.pdf).

For formatting the required file, see tip_info_tt.txt in the tropical_trees folder containing the tip information for the tropical trees.

d. Bootstrap

i. Nonparametric vs. parametric bootstraps

Nonparametric bootstrapping can be computationally intensive when the dataset is large (>100G in total). If you think it takes too long, switch to the parametric one. Also with large genomes, the bootstrap value tends to stay as 100%.

ii. Correction factor

Set to be 2. The correction factor in the R script is from Equation 11 that estimates the variation caused by sequencing error and incomplete coverage. Detailed simulations showed that the formula sometimes under-estimates and sometimes over-estimates the true standard deviation in the distance between taxa by as much as 50%. This occurs because of the complexities of accounting mathematically for the correlations among *k*-mers that occur on the same reads. The correction factor set to 2 provides a conservative bootstrap (i.e., one that is not going to improperly inflate the support for nodes), be insuring that the true variation caused by sequencing error and incomplete coverage is never

underestimated.

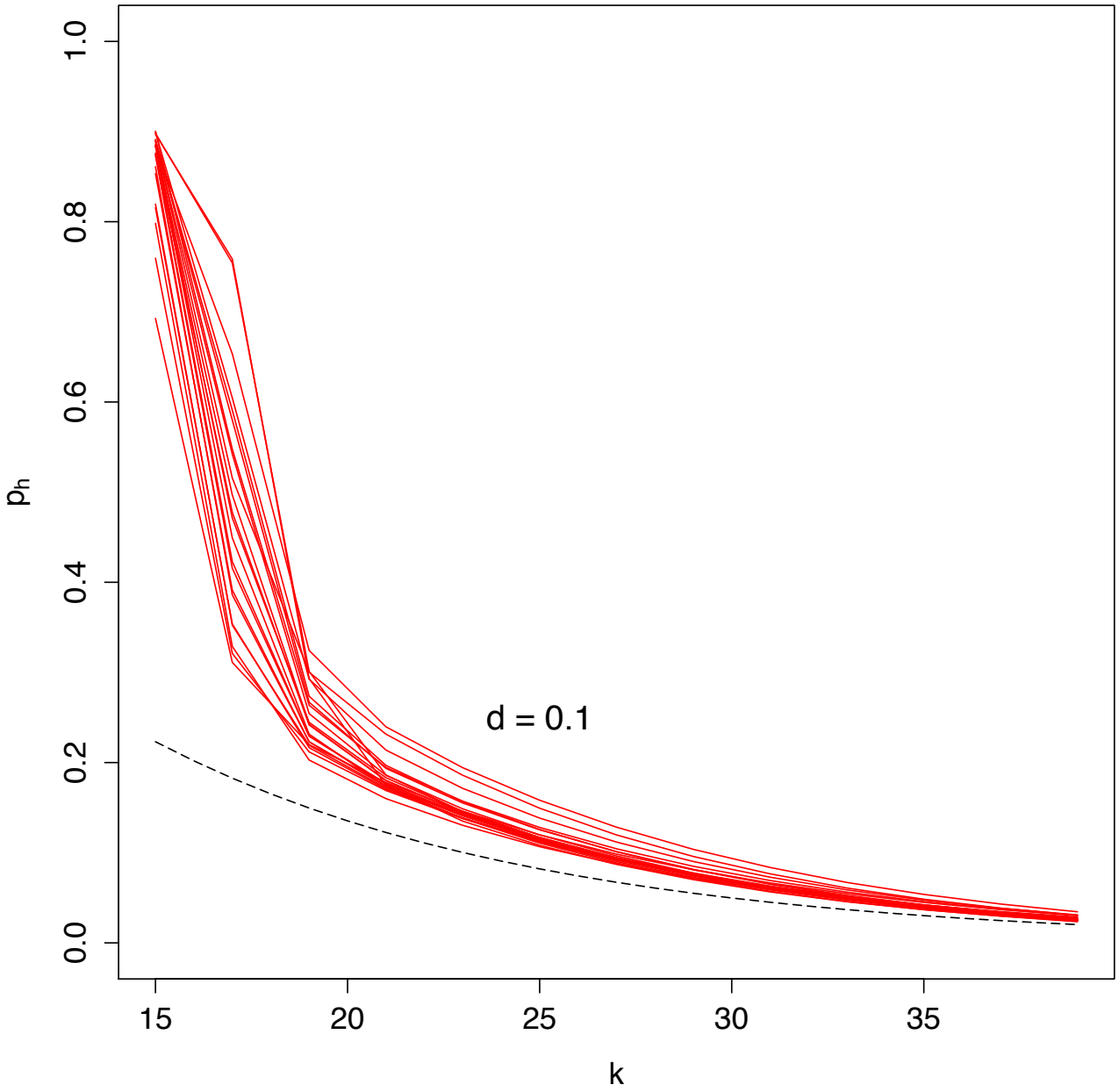

Figure S6. Theoretical predictions of the proportion of shared *k*-mers, p_h, calculated from the observed frequency distribution of *k*-mers, Q_k, for the tropical trees dataset ranging in genome size from 250M to 2Gbp assuming the true distance between taxa is $d = 0.1$ (divergence time 94Mya).

Reference:

Marcais G, Kingsford C. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* **27**: 764–770.