**kSNP4 Users Guide**

Barry G. Hall, Bellingham Research Institute

Jeremiah Nisbet

and

Shea N. Gardner (Deceased), Lawrence Livermore National Laboratory

Download either the **kSNP4 Mac Package** or the **kSNP4 Linux Package**. Each package contains a **kSNP4pkg** directory and a Documentation directory (folder) that includes this User Guide. Also download the two example directories, each of which contains input files for the examples discussed in section VIII. Installation of ***kSNP4*** is discussed in section II.

Table of Contents

[I. The kSNP4 program 3](#_Toc112704889)

[II. Installing kSNP4 3](#_Toc112704890)

[Set the PATH variable 4](#_Toc112704891)

[III. Obtaining the genome sequence files 5](#_Toc112704892)

[Organizing your genome files 5](#_Toc112704893)

[IV. Input files 6](#_Toc112704894)

[File names 6](#_Toc112704895)

[V. Making the input file for kSNP4 7](#_Toc112704896)

[VI Using Kchooser4 to determine optimal value of k, the kmer size 8](#_Toc112704897)

[VII. Running kSNP4 10](#_Toc112704898)

[Annotating genomes 11](#_Toc112704899)

[Capture a Logfile 12](#_Toc112704900)

[Other arguments 12](#_Toc112704901)

[kSNP4 tree accuracy 12](#_Toc112704902)

[Labeling tree nodes with imperfect but significant SNPs associated with the node 12](#_Toc112704903)

[VIII. Tutorials with Examples 13](#_Toc112704904)

[VIIIA Tutorial: Example 1: Only finished genomes 13](#_Toc112704905)

[VIIIB Tutorial Example2: Finished and unfinished genomes 14](#_Toc112704906)

[IX The output files 15](#_Toc112704907)

[X. The kSNP4 Utilities 20](#_Toc112704908)

[Utilities in alphabetical order 20](#_Toc112704909)

[checkGenbankFromNCBI 20](#_Toc112704910)

[extractNthLocus4 20](#_Toc112704911)

[fixOldFastaHeaders 21](#_Toc112704912)

[genomeNames4 21](#_Toc112704913)

[Kchooser4 21](#_Toc112704914)

[MakeKSNP4infile 21](#_Toc112704915)

[NodeChiSquare2tree4 21](#_Toc112704916)

[rmNodeNamesFromTree4 21](#_Toc112704917)

[selectNodeAnnotations4 22](#_Toc112704918)

[fixOldFastaHeaders 22](#_Toc112704919)

[XI How kSNP4 works 22](#_Toc112704920)

[XII Citations 23](#_Toc112704921)

[XIII Licenses 24](#_Toc112704922)

[XIII FAQ (Frequently Asked Questions) 25](#_Toc112704923)

[Appendix I Suggested text editors 28](#_Toc112704924)

[Appendix II Details of Downloading genome sequences and preparing an input file. 29](#_Toc112704925)

kSNP4 is a command-line program to compare genomes sequences without aligning them. If you are unfamiliar with the command-line interface please read the document "Command-line Environment Mac-Linux.pdf" that is in the Documentation directory.

# I. The kSNP4 program

Single nucleotide polymorphisms are important for phylogenetic analysis, for tracking viral and bacterial pathogens during outbreaks, and for correlating genotype with phenotype. When dealing with gene *sequences* the first step in identifying SNPs is multiple alignment of the gene sequences.

Bacterial and viral genomes are characterized by multiple and massive insertion-deletions (indels), inversions, and transpositions of blocks of DNA from one part to another part of the genome. The term "pan-genome" has been coined to describe the collective content of the genomes of a bacterial species. The pan-genome consists of core genes, those genes that are present in all members of a species, and accessory genes, those that are present in some, but not all, members of the species. Indeed, there is at least as much variation in the presence/absence of DNA sequences as there is variation in SNPs. That variation makes multiple sequence alignment of complete genomes impractical for large numbers of genomes.

kSNP4 is a program that identifies the pan-genome SNPs in a set of genome sequences, and estimates phylogenetic trees based upon those SNPs. Because there are many potential downstream applications of SNP information, kSNP4 also provides output files that include a multiple alignment of all of the SNP positions and files that provide information about the positions of each SNP in each genome and annotations. kSNP4 can analyze both complete (finished) genomes and unfinished genomes in assembled contigs or raw, unassembled reads. Finished and unfinished genomes can be analyzed together, and kSNP4 can automatically download Genbank files of the finished genomes (and annotated genome assemblies) and incorporate the information in those files into the SNP annotation. Core SNPs, those present in all of the genomes, can optionally be analyzed separately to generate a multiple SNP alignment and trees based on only the core SNPs. kSNP4 also provides a separate analysis of those SNPS that occur in at least a user-determined fraction of the genomes.

kSNP4 is provided as both a Linux package for 64-bit CPUs and as a Mac OS X package for 64-bit CPUs. The packages contain the kSNP4 program and this documentation. In addition there are two example input data sets that are used as the basis for the discussion of the output files in Section III. Those data sets must be downloaded separately from SourceForge.

kSNP4 incorporates six third-party programs. Please see section VI, Citations, for the list of publications describing these software packages.

kSNP4 is a command-line program that must be run from the Terminal application. If you are unfamiliar with the command-line interface please read the document "The Command-Line Environment" before attempting to install or use kSNP4.

# II. Installing kSNP4

A note about terminology: Linux users use the term "directory" while Mac users use the term "folder" to mean exactly the same thing. Throughout this User Guide we will use the term "directory". Mac users, when you see "directory" think "folder".

kSNP4 consists of a Unix shell script and a set of executables that includes four separate third-party programs to accomplish the kSNP4 goals. kSNP4 is provided as separate packages for Mac OS X and for Linux operating systems. Whichever version you chose was downloaded as a zip archive and should automatically self-extract. If it does not, double-click the file to extract it. In the resulting package directory you will see a directory named kSNP4pkg that contains some 61 files. The kSNP4pkg directory contains a bash script named kSNP4 that directs the execution of a set of unix executable perl and python scripts, including the 3rd party programs jellyfish, FastTreeMP, parsimonator, mummer, sa and consense. You do not have to deal with most of those files directly, they are called by kSNP4 scripts.

Warning!! You are strongly encouraged to install one of the packages of executables and not to try to install the source code. The only reason to download the source code is if you need to modify that code for your own purposes.

If you work from the source code no support will be provided. You are entirely on your own. If you modify the source code you are again on your own and no support will be provided

kSNP4 should be installed into the /usr/local directory, and once installed the kSNP4pkg directory should not be moved afterwards.

The /usr/local directory is protected so you need to make some extra effort to get access to it and to add things to it.

**Mac OS X:** in the Finder, from the Go menu choose Go to directory… and in the resulting dialog box enter /usr/local. The local directory will open. Drag the kSNP4pkg directory from the kSNP4\_Mac\_package directory into the usr/local directory. You may be notified that kSNP4pkg can't be moved because local can't be modified. Don't worry. Just click the **Authenticate** button and enter the Administrator's password. (If you aren't an administrator find the person who is and get them to authenticate for you).

**Linux:** In Terminal navigate to the kSNP4\_Linux\_package directory that contains the kSNP4pkg directory then enter sudo cp kSNP4pkg /usr/local. You will be asked for a password. Enter the Administrator's password. (If you aren't an administrator find the person who is and get them to authenticate for you).

To check that the package was actually copied to /usr/local enter cd /usr/local, then enter ls to list the contents of /usr/local. You should see kSNP4pkg listed. If you do not, try again or get help from your computer's Administrator.

### Set the PATH variable

The operating system needs to know exactly where all the programs in kSNP4pkg are located when it calls them to action. To do that you must set the PATH environmental variable. A path is just a description of the nesting of directories starting with the root. Directory names are separated by /. The path /usr/local means the local directory that is in the usr directory that is at the root. The PATH variable usually includes paths to several default directories where unix programs may be located, and you can add paths to other directories where you have installed other unix programs. To see the paths that are currently available, enter echo $PATH.

If the path variable has never been modified you will see something like this (there may be other paths as well):

/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

To add paths you must modify a file that holds environmental variables. Under the bash shell that file is called .bash\_profile and it is found in your home directory. Under the zsh shell (the default shell in some recent versions of both Mac OS and Linux operating systems) the file that holds environmental variable is called .zshrc In both cases the file is invisible because its name begins with a dot.

In **Linux** type control-H, or in your home directory under the **View** menu choose **Show hidden files**. In **Mac OS X** type Command-shift-period. You should now see the hidden files. If you do not see a .bash\_profile or a .zshrc file don't worry. Just create the file in a text editor (see Appendix I for suggested text editors). Don't forget the leading dot in the file name.

The instructions below assume that you installed kSNP4pkg into usr/local. If you installed kSNP4pkg elsewhere add the path to the installed location instead.

If the .bash\_profile or .zshrc file does not exist, make a new text file named .bash\_profile or .zshrc in your home directory (don't forget the leading dot!), then enter the following line *exactly as given here; i.e. no spaces around the = sign*

export PATH="/usr/local/kSNP4pkg:$PATH"

If the .bash\_profile or .zshrc file already exists just add that line to it.

In both cases save the file in your home directory, then close the Terminal window. A Terminal window only becomes aware of the available paths when it is first opened.

# III. Obtaining the genome sequence files

kSNP4 works on genomes sequence files in the fasta format. Those files must be stored on your hard drive in directories. It is important to put the files into appropriate directories and not to move them after making the input file for kSNP4 (more about that in the next section). I organize my genome sequence files in a directory named Genomes, within which is a directory named Bacteria and another directory named Viruses. Within each of those directories I have a directory for each species; e.g. E. coli, Salmonella, etc. You can use any organization that you wish, but it is a good idea to be consistent.

### Organizing your genome files

We suggest collecting all of the .fasta files into a single directory that is named for the species and *putting that directory somewhere from which it will never be moved*. The best strategy might be to have a directory named, for instance, "Genomes" Within that you might have a directory named "E\_coli", another named "S\_aureus", etc. If you wanted to keep annotated genomes separate from unannotated genomes, within each species directory you could have directories "Annotated" and "Unannotated".

Not only must you never move the individual directories or the "Genomes" directory *you must never move any of the directories that enclose those directories!!* If you move a directory or an enclosing directory then the paths in any existing kSNP4 input file will be wrong and kSNP4 won't be able to find the genome files - nothing will work. You would have to reconstruct every affected input file.

The solution is to put the Genomes directory into your home directory and to leave it alone. Do ***not*** put your Genomes directory onto another Volume, such as an external hard drive. kSNP3 won't be able to find the files.

A common application of kSNP4 is to compare a set of genome sequences that one has made from a collection of isolates of interest with known genome sequences that are in NCBI databases. Often that comparison involves making a phylogeny that includes both NCBI's genomes and the lab genomes.

It can be very tedious to collect all the genome sequence files from NCBI, so I have created two programs to assist with finding and downloading genome sequence files: ***parseNCBIcsv*** and ***getFastaGenomes***. Their use is detailed in a separate document "Downloading Genomes from NCBI.pdf" that is in the Documents directory of the kSNP4 package that you downloaded. Both the ***parseNCBIcsv*** and ***getFastaGenomes*** executables are in the kSNP4pkg folder that you installed.

# IV. Input files

All of the programs in the kSNP4 package require input files. Those input files are simple text files. Word processor files written by Microsoft Word, WordPerfect, etc. will not work. Input files should be opened in a text editor. See Appendix I for suggested text editors

### File names

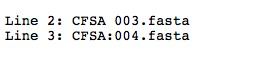
It is important that the input files are correctly named. Depending on your operating system, file names that violate the rules below may cause horrible results - including greatly inflating the number of SNPs, or failing to annotate the SNPs correctly. kSNP3 may appear to have run properly and unless you have had considerable experience with similar data sets you may not recognize that errors have occurred. To avoid that situation kSNP3.03 and later versions check the input file for file names that violate the rules and abort if any files are incorrectly named.

A file name consists of two parts: a file ID and an extension. The extension is separated from the file ID by a dot. For the file named **Escherichia\_coli\_042.fasta**, the file ID is Escherichia\_coli\_042 and the extension is fasta.

The rules for naming files are:

1. The file name cannot include more than one dot (.); i.e. only the dot separating the file ID from the extension is allowed. Escherichia\_coli\_042.fasta is legal; Escherichia\_coli\_0.42.fasta is not. (It is probably leagal for your operating system, but it is not legal for kSNP4).
2. The file name may not contain any spaces. Escherichia\_coli\_042.fasta is legal; Escherichia coli 042.fasta is not.
3. The file name can only contain the characters A-Z a-z 0-9 . - (the dash or minus character) and \_ (the underscore character). Characters such as : \* > are illegal. When necessary for readability replace spaces, colons etc. with the underscore character or use internal capitalization. For instance Ecoli\_O157H7 instead of Ecoli O157:H7.

If kSNP4 finds a file with an illegal character it terminates the run and writes a file named NameErrors.txt. NameErrors.txt reminds you of the naming rules and writes a list of the files that contain naming errors. Part of that file might look like this:



Each error includes the line in the input file where the error occurred and the illegal file name. On Line 2 the file name is illegal because name includes a space. On Line 3 the file name is illegal because it includes a colon.

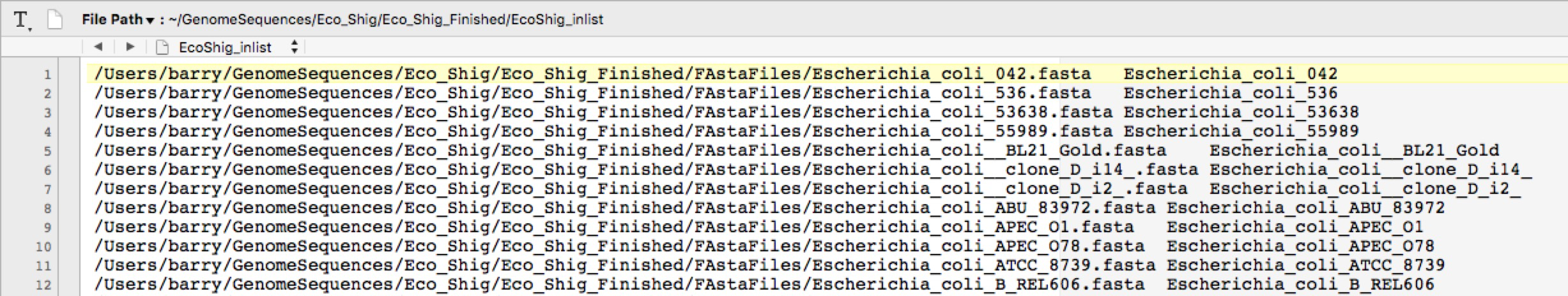
Correct the illegal file names on the files themselves and in the input file!!! Once that is done throw the NameErrors.txt file into the trash. That is essential because the way that kSNP4 knows to terminate the run is by looking for a file named NameErrors.txt. If it finds that file, even if it a left-over file, kSNP4 will terminate the run.

# V. Making the input file for kSNP4

A data set consists of a set of a set of genome sequences in fasta format, one file per genome. **Note! Raw-read files in the fastq format will not work. See the FAQ about Fastq in the FAQ section)** The genome can be a finished (closed) sequence, multiple chromosomes and plasmids, an assembly of multiple contigs, or raw, unassembled reads.

The input file is a list that gives the *path* to each sequence file containing a genome and a name for that genome, with each genome on a new line. The input list tells kSNP4 where to find each genome sequence file.

The input file might look like this:



Each line consists of a path separated by a tab from the genome ID. The genome ID will be used in all of the trees and other output files. The genome ID on the first line is Escherichia\_coli\_042, while the *file name* is Escherichia\_coli\_042.fasta.

The easiest way to create the input file is to use the program ***MakeKSNP4infile*** in either the fully automatic mode (easiest method) or the semi-automatic mode.

Fully automatic method

Navigate to the directory that *encloses* the directory containing the fasta genome files and enter MakeKSNP4infile myDir myInfile A. myDir is the name of the directory that contains the fasta files. myInfile is the name for the kSNP4 input file that will be written, and A indicates the fully automatic process in which the genome ID is taken from the file name (any extensions, such as .fasta, are dropped).

Note: The automatic mode is useful only when the fasta file name is the genomeID that you want to appear in all the phylogenetic trees.

Semi-automatically

Navigate to the directory that *encloses* the directory containing the fasta files and enter MakeKSNP4infile myDir myInfile S. myDir is the name of the directory that contains the fasta files. myInfile is the name for the kSNP3 input file that will be written, and S indicates the semi-automated process in which the paths to each of the files are written followed by a tab, but you must manually enter the genome ID. This method is useful when the names of the fasta genomes are too long to fit easily onto a drawing a tree.

In both cases you can edit the genome ID in the infile at any time.

Manually

In a text file enter the path to the genome file, press Tab, then type the name for the genome. When there are many genomes this can be a tedious process.

You certainly don't want to have to figure out and type the entire path (/Users/barry/Desktop/Test\_kSNP3/Genomes/ Escherichia\_coli\_042.fasta), but there is an easier way to enter the path depending upon your operating system:

In Mac OS X hold down the Command key and drag the file into the text document. The path will magically appear.

In Ubuntu Linux (and several other Linux versions, I understand) select the file, right-click and in the resulting menu choose Copy. Paste into the text document and the path will magically appear.

For other versions of Linux you will need to figure out how to get the path and enter it into the text file.

# VI Using Kchooser4 to determine optimal value of k, the kmer size

kSNP4 requires that you enter the kmer size on the command line using the -k option. kmer size, *which must be* *an odd number*, defines the length of the oligonucleotides (kmers) that kSNP4 identifies in all of the sequences. Oligos that are identical between different genomes at all but the central base are taken as being homologous, i.e. a SNP locus. If the kmer size is set too low, say to a value of 5 bp, then there will be many such kmers that are identical by chance alone within a genome and between genomes, rather than being identical by descent from a common ancestor; i.e. homologous. If the target genomes are short, then the chance of spuriously identical kmers is reduced; whereas long genomes increase the chance of spurious matches. That consideration would seem to favor choosing large values for kmer size. However, if kmer size is set too high, say 51 bp, then because of sequence variation at multiple sites within the kmer, many SNPs will be missed because a SNP locus is defined by the conserved sequence surrounding the central base of the kmer. When there is little base-pair variation large values decrease the chance of spurious matches, but when there is much variation large values decrease the sensitivity of SNP detection. The program ***Kchooser4*** identifies the optimum value of k for your particular data set.

***Kchooser4*** first identifies an optimum value of k for your specific data set, then it evaluates the extent of sequence variation to provide insight into the efficiency with which kSNP4 is likely to identify SNPs from your data set and the accuracy of parsimony trees at that optimum k.

To estimate the optimum k, ***Kchooser4*** begins with a k value that is a function of the median genome size. It determines FUK, the fraction of unique kmers, i.e. those that occur only once in the median-length genome at that value of k. Some kmers will occur more than once because of genuine duplicated regions, so ***Kchooser4*** continues seeking a value of k such that at least 0.99 of kmers are unique. If FUK is < 0.99 it increments k by 2 and tries again. If the fraction is ≥ 0.99 it reports that as the optimum value of k. The expectation is that less than 1% of a real genome will be duplicated.

In some cases more than 1% of the genome is duplicated, so ***Kchooser4*** would continue increasing k forever. However, if more than 1% of the genome is duplicated, FUK approaches the fraction of the genome that is actually unique. When FUK increases by <0.001 ***Kchooser4*** reports the current value of k as the optimum value.

***Kchooser4*** then determines what fraction of the kmers from the shortest sequence are present in all of the genomes; i.e. the fraction of core kmers (FCK). As sequence variation increases FCK decreases and the efficiency with which kSNP4 detects SNPs decreases. Simulation studies (Gardner & Hall 2013 PLoS One **8**(12): e81760. doi:10.1371/journal.pone.0081760, Hall 2015 Cladistics **32:** 90-99) have shown that SNP detection efficiency is >97%, and the accuracy of parsimony trees is > 0.97 when the fraction of core kmers reported by ***Kchooser4*** is ≥ 0.1.

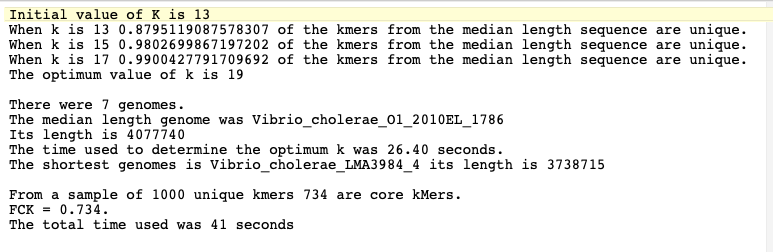
When ***Kchooser4*** is complete it writes a report file whose name is based on the name of the input file.

The input to ***Kchooser4*** is the input file for kSNP4 that you created in Section IV using the program ***MakeKSNP4infile***.

Usage: Kchooser4 -in infileName

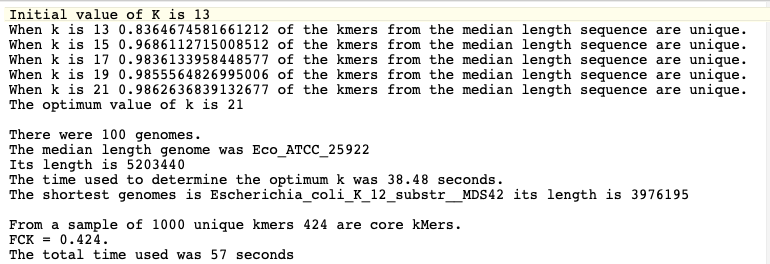
Example: Kchooser4 -in VC.in.

Output file: Kchooser4\_VC.report.



FUK was 0.99004 when k was 17 so the optimum value of k is 17 + 2 = 19.

With a different data set the input was BigData.in.



The fraction of unique kmers has reached a plateau at a bit over 0.98, suggesting that a bit more than 1% of that particular genome is duplicated. Increasing k from 19 to 21 increased FUK by < 0.001, so ***Kchooser4*** reports an optimum k of 21.

***Kchooser4*** runs fairly quickly. For a data set consisting of 100 *E. coli* genomes ***Kchooser4*** required 72 seconds. The inclusion of genomes as raw reads can dramatically increase the time required by ***Kchooser4***. To speed up the process raw read genomes should be removed from the input file before running ***Kchooser4***.

# VII. Running kSNP4

kSNP4 is run from the command line within the ***Terminal*** program.

Enter the word kSNP4 followed by a series of *arguments* separated by spaces. Arguments are instructions that tell ***kSNP4*** exactly what to do. An argument consists of a flag and a value, for instance the argument

-in in\_list says that the input file is named in\_list. There must always be a space between the flag and the value, and there must always be a space between an argument and the next argument. The table below lists the arguments and what they mean. Arguments may be entered in any order.

Table 1 Command line arguments\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Argument** | **flag** | **value** | **example** | **comment** |
| input file listing paths and genome names of genome fasta files for analysis | -in | file name | -in infileName | required |
| directory for output files | -outdir | name of directory | -outdir Run1 | required |
| k-mer size | -k | odd integer | -k 13 | Required, length of kmer containing the SNP |
| file of genome names to use as references for gene annotation of SNPs | -annotate | file name | -annotate annotatedGenomes | optional. List of names of genomes to use for annotation. |
| calculate core SNPs and core parsimony tree | -core | no value | -core | Optional  Calculate loci and tree based only on the SNPs found in all genomes |
| minimum fraction of genomes with locus | -min\_frac | decimal fraction between 0 and 1 | -min\_frac 0.5 | optional  Calculate a tree based on only SNP loci occurring in at least this fraction of genomes |
| Number of CPUs to use | -CPU | integer between 1 and the number of CPU’s available | -CPU 12 | optional,  defaults to the number of processors available |
| Estimate Neighbor Joining tree | -NJ | no value | -NJ | optional. calculates an NJ tree. Default is to not calculate an NJ tree |
| Estimate maximum likelihood tree | -ML | no value | -ML | optional. calculates an ML tree. Default is to not calculate an ML tree |
| Generate VCF (Variant Call Format) files | -vcf | no value | -vcf | optional  Generates VCF files. Default is to not generate VCF files. |

\* file and directory names must not contain any spaces and should avoid any special symbols such as ()\*&%#@{}[]. Case matters. -nj is not the same as -NJ! See Section IV for the rules about naming files.

**Example:**  kSNP4 -in Eco100.in -k 21 -outdir Run1 -annotate annotatedGenomes

The command line says that ***kSNP4*** is to use Eco100.in as the input file, the kmer size is set to 21, and the output files will all be written in the Run1 directory. In addition to those required arguments ***kSNP4*** is to annotate the genomes using a list of genome names in the file annotatedGenomes.

## Annotating genomes

It can be very useful to know something about where each SNP is found in a genome, what gene product (if any) the SNP lies within, whether the SNP results in a synonymous or non-synonymous changes, etc. ***kSNP4*** retrieves that information from NCBI by downloading the fully annotated version of a genome file and using the information in those annotations to annotate each SNP. The command to annotate (-annotate) must be followed by the name of a file that lists the genomes to use for annotation, annotatedGenomes in the above example but you can name the file anything you wish.

To easily make a list of file name run the program ***genomeNames4***. The command is:

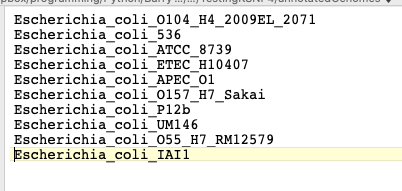
genomeNames4 infileName outfileName, where infileName is the name of the input file you used for ***kSNP4*** and outfileName is the name of the file listing the genomes to use for annotation.

Example: genomeNames4 Eco100.in annotatedGenomes.

It is important to only list genomes that you know are annotated in the corresponding NCBI .gbk file. Typically you only want a representative sample of the genomes that are in the ***kSNP4*** input file. I usually try to choose no more than 10% of the genomes for annotation.

It is important to put the most reliably annotated genomes at the top of the list; i.e. finished genomes before annotated genome assemblies. SNPs are annotated with reference to the first genome in the list in which the SNP is present. Do not include genomes available only as raw reads in the annotated\_list! Only include those genomes for which you want to know the position of the SNP. Otherwise, it will waste time finding and reporting positions for every read that covers the SNP, which can result in a massive SNPs\_all.

In the above example annotatedGenomes would list all 100 genomes, so I would go through the list deleting names until I was down to about 10 genomes I thought were representative. The final annotatedGenomes file might look like this:



### Choosing the genomes to use for annotation

Every genome that is in the annotatedGenomes list must be a genome for which the annotations are available at NCBI in a .gbk file. You can assume that all complete genomes are full annotated. Some that are not complete, but instead are available as scaffolds will be annotated, while those that are only available as contigs may or may not be annotated. The safest choices are closed, complete genomes.

As pointed out above, you want a representative set of genomes for annotation, but not more than 10 or 20 or so genomes if possible. The problem is how to choose the representative genomes.

What you want is to have one good representative from each group of fairly closely related genomes; i.e. from each clade. In order to judge that you need to actually look at a drawing of the phylogenetic tree, so step 1 is to run kSNP4 *without* annotation, export the parsimony tree to a tree drawing program such as MEGA X or FigTree. Then look at that drawing and try to identify the groups that you want to be represented.

The figure below shows the parsimony tree of 47 genomes in radiation format. Those genomes include 17 unfinished genomes, those whose name begins with Eco-\_I\_ .The various groups, as I see them, are enclosed in colored shapes.

The 6 groups enclosed in a colored shape represent fairly coherent clades. The object of this exercise is to identify, within each group, one finished genome that can be use for annotation. I chose MB10 (violet), YD786 (green), A15 (yellow), UTI89 (blue), AR\_0089 (red) and P31\_YH1\_02\_21 (turquoise)I saved all of those to the file annotatedGenomes.txt and ran kSNP4 with annotation (Run 2)

## Capture a Logfile

***kSNP4*** generates a lot of screen output as it runs, including the time used in the run. It can be very useful to have a record of that output, for instance to compare run times for different data sets. To capture the screen output as a log file use "tee" to send the output to both the screen and a file. In the bash shell the command is implemented by adding | tee Logfile.txt after the usual kSNP3 commands, for example

kSNP4 -in Eco100.in -k 21 -outdir Run1 -annotate annotateGenomes | tee Run1Log.txt

(What about the zsh shell? Does | tee work?)

## Other arguments

**minimum fraction with locus:** Because of indels or sequence variations within the kmer around a SNP, many SNPs will not be present in all of the genomes, and some may be present in only two genomes. In addition to analyses based on all of the SNPs and analyses based only on those that are present in all genomes (core SNPs), it is sometimes useful to also base a SNP analysis only on the SNPs that are present in some minimum fraction of the genomes . This can be set to any decimal fraction between 0 and 1.0. Note that 0 is the same as all SNPs, and 1 is the same as core SNPs.

## kSNP4 tree accuracy

Simulation studies (Hall, 2015 Cladistics **32:** 90-99) have shown that kSNP3 parsimony tree are the most accurate, followed by ML trees, with NJ trees being the least accurate. For that reason the default is to only estimate parsimony trees. The parsimony tree that is estimated is a consensus of up to 100 equally parsimonious trees.

## Labeling tree nodes with imperfect but significant SNPs associated with the node

The tree\_AlleleCounts.X.tre output files are phylogenetic tree files in which the nodes are labeled with the number of SNPs that are present in all of the descendants of that node and nowhere else (see Table 4, below). Think of those SNPs as being perfectly associated with that node. In many cases there are nodes that have no perfectly associated SNPs, but nevertheless do have SNPs that are imperfect but significantly associated with that node. The utility ***NodeChiSquare2tree4*** identifies those SNPs and writes a tree file in which the nodes are labeled with the number of those significantly associated SNPs. (See section VI, The kSNP3 Utilities, under Utilities to use after running kSNP3).

***NodeChiSquare2tree4*** is run from within the kSNP3 output file directory after kSNP4 has run. Decide if you want to run ***NodeChiSquare2tree4*** after examining the tree\_AlleleCounts.X.tre file of interest.

# VIII. Tutorials with Examples

Because ***kSNP4*** was designed to maximize flexibility, and in anticipation of future uses of comparative SNP information, ***kSNP4*** writes a plethora of output files. Indeed, analysis of just 10 viral genomes produces up to 73 output files.

Note! You should download the two example input files that are discussed below. Going through those examples will not only familiarize you with ***kSNP4***, it will allow you to determine whether you have installed ***kSNP4*** correctly.

Note! Please do not copy the command line entries from this document. The dashes that precede the flags may be converted to em-dashes in this document, in which case nothing will work. Instead, enter the commands manually or copy them from the CommandLines.txt file in each example.

## VIIIA Tutorial: Example 1: Only finished genomes

The first example is one in which the data set consists of finished genome sequences. The Example1 directory contains three items: (1) the data set within a directory named Genomes. That data set consists of finished sequences of 10 encephalitis virus genomes. (2) a file named CommandLines.txt and (3) a directory named "Results" The purpose of that directory is to allow you to compare your results form this tutorial with my results. Within that directory there are the input files Example1.in and annotatedGenomes. ***You cannot use the file Example1.txt for input to your kSNP4 program!!!*** The paths in that file are not correct for you because the Genomes directory is somewhere on your hard drive and the paths point to the directory on Barry Hall's machine There are also the results of Runs 1, 2 & 3 done by BGH. All of that is for you to compare with the input and output files that you generate during this tutorial.

**Step 1: Generate the input file and run Kchooser4**

1. Generate the input file list by running ***MakeKSNP4infile***. Navigate to the Example1 directory and enter MakeKSNP4infile Genomes Example1.in A . The result will be an input list file Example1.in.
2. Run ***Kchooser4*** to determine the optimum kmer length and to estimate FCK. Enter Kchooser4-in Example1.in. The output file will be named Kchooser4\_Example1.report. It tells us that the optimum k is 13, and FCK = 0.05. Since FCK is < 0.1 the accuracy of the tree is probably not very high and we should be cautious about its interpretation.
3. Generate a list of annotated genomes. Since all of the genomes are annotated you can extract the list from Example1.it. For the sake of this tutorial call the output file annotatedGenomes, but you could call it anything you like. Enter genomeNames4 Example1.in annotatedGenomes.

**Step 2: Run kSNP4**

2A **without annotations**

The simplest way to run kSNP4 is without doing any SNP annotation. We will set the kmer size to 13 as suggested by ***Kchooser4***. We will call this run Run1. All the options except -in, -outdir, and -k take their default values.

Enter kSNP4 -in Example1.in -outdir Run1 -k 13 | tee Run1Log

The 17 output files will be found in the Run1 directory. At the bottom of the Run1Log file you will see that on my computer the run required 0.025 hours, or 91 seconds. Note that in column 4 of the SNPs\_all file, they are all x (unknown).

2B **with annotations**

To annotate the SNPs we simply add the -annotate option, specifying the annotated\_genomes files that lists the annotated genomes.

In this case all 1of the genomes are annotated so annotated\_genomes lists all of the genomes. More typically a data set would include some finished genomes, some assembled genomes that are annotated, some assembled genomes that are not annotated, and perhaps some raw read files. In that case you would delete any unannotated and raw read genomes from the annotated\_genomes list file.

Enter kSNP4 -in Example1.in -outdir Run2 -k 13 -annotate annotatedGenomes | tee Run2Log

Because of annotation you will find 22 files in the Run2 directory. Annotation increased the run time to 129 seconds on my computer.

The annotations output files you want to look at are "SNPs\_all", "Annotation\_summary", and "SNPs\_all\_annotated". See Table 3 for the contents of those files

2C **with all options used**

Finally, we can add the options to estimate ML and NJ trees, to generate VCF files, to calculate core SNPs to estimate a core parsimony tree, and to set the minimum faction with locus to 0.75.

Enter kSNP4 -in Example1.in -outdir Run3 -k 13 -annotate annotatedGenomes -ML -NJ -vcf -core -min\_frac 0.75 | tee Run3Log

The Run3 directory will now include 69 files, and the run on my computer required 136 seconds

The purpose of these multiple runs was to illustrate that the more you ask kSNP3 to do, the longer it takes. It is also the case that the more genomes you include in the data sets, and the larger those genomes are, the longer it takes.

## VIIIB Tutorial Example2: Finished and unfinished genomes

Before discussing this example, a quick word about unassembled genomes. Before running ***kSNP4***, we advise a first step of filtering/trimming low quality bases or reads from the fastq using something like fastq\_quality\_trimmer or seqtk before creating the fasta file of reads. This will help avoid considering sequencing errors as SNPs.

This *Vibrio cholera*  data set includes three finished genomes (VcMS6.fa, VcLMA3984-4.fa and VcO1-ElTorN16961.fa, each of which consists of two chromosomes), two assembled genome that consists of a set of contigs (Vc523-80.fa and Vc63-93\_MO45.fasta), and an unassembled genome that consists of **raw reads** that has been filtered and trimmed (ERR579925.fasta). You should notice that the raw reads file is *huge*, 617 megabytes, compared with the other files that are about 4 MB.

The purpose of this example is four-fold: (1) to illustrate the use of a mixture of genome types, (2) to point out how ***kSNP4*** handles and annotates multi-chromosome genomes, (3) to illustrate the use of ***Kchooser4*** to determine the optimum kmer size with bacterial genomes.

The Example2 directory contains the same items as does the Example1 directory

Begin my navigating to the Example2 directory and making a ***kSNP4 / Kchooser4*** input file just as you did for Example1. The command line is MakeKSNP4infile Genomes Example2.in A.

Next run ***Kchooser4*** using the command line Kchooser4 -in Example2.in. The output file will be named Kchooser4\_Example2.report. The optimum value of k is 21, and FCK = 0.724. FCK is well above 0.1, so the trees estimated by ***kSNP4*** should be quite accurate and > 97% of the SNPs should be found.

Next, make a list of the genomes to use for annotation. Enter genomeNames4 Example2.in annotatedGenomes. The annotatedGenomes file will list all six genomes, but we know that only VcMS6.fa, VcLMA3984-4.fa and VcO1-ElTorN16961.fa are complete and should be used for annotation. Open annotatedGenomes in a text editor and delete the lines that are not complete genomes. Be sure to delete the entire line in each case, leaving only 1 blank line at the end of the file.

Finally, after all that, run ***kSNP4*** by entering:

kSNP4 -in Example2.in -outdir Run1 -k 21 -annotate annotatedGenomes | tee Run1Log

On my computer Run1 required 20.4 minutes.

# IX The output files

Most of the output of ***kSNP4*** is presented separately for three groups of SNPS:

(a) ***all*** SNPs

(b) the ***core*** SNPS, which are the SNPs that are present in all of the genomes

(c) the ***majority*** SNPs, which are SNPs that are present in the user-defined minimal fraction of genomes (see option –min\_frac in Table 1). The cutoff for inclusion in the majority is incorporated into the name; i.e. majority0.5 means that only SNPs in at least 0.5 of genomes are included. ***Core*** and ***majority*** SNPs are subsets of ***all*** SNPs.

The output files can be grouped into several categories:

(1) alignment, or *matrix*, files that contain alignments of various groups of SNPs

(2) files that give information about the SNPs, including position of the SNPs in the genomes and annotation information

(3) files that give phylogenetic trees based on all, core, or majority SNPs created using several methods

(4) files that give counts of SNP alleles shared by various subsets of genomes

(5) files that give the number of SNPs that correspond to nodes on the trees

(6) files that give information about homoplasy groups

(7) files that give information for each locus about the node or homoplasy group to which it belongs

(9) other files

Table 2. The alignment files

|  |  |
| --- | --- |
| **File name** | **Explanation** |
| SNPs\_all\_matrix  core\_SNPs\_matrix  SNPs\_in\_majority0.5\_matrix | Relaxed PHYLIP format of the SNP alleles. Genome name - SNP allele string  The SNP alleles for each genome are concatenated into a string whose length is the number of SNPs. N indicates that the SNP is absent in that strain. Loci are concatenated in the same order as listed in the SNPs\_all file. |
| SNPs\_all\_matrix.fasta  core\_SNPs\_matrix.fasta  SNPs\_in\_majority0.5\_matrix.fasta | >Genome name  SNPs string  The same information as above, except in fasta format |

The fasta alignment files can be thought of as the "master" files of SNPs. Fasta is the most common input format for phylogenetics software, and can be used to estimate either phylogenetic or minimum spanning trees using your favorite software. Likewise, the fasta files can be used for other purposes such as phenotype association studies.

Table 3. Files that provide information about the SNPs

|  |  |
| --- | --- |
| **File name** | **Explanation** |
| SNPs\_all\_annotated | Provides the most complete information about each SNP. Each SNP is given a locus number, and a particular SNP is listed once for each genome in which it occurs. Information followed by an asterix is provided only for annotated genomes. The information includes the SNP base in that genome, the genome name, the accession number\*, the position of the SNP in the genome\*, the codon in which it occurs, the amino acid encoded (in protein coding regions), the residue peptide context\*, and the gene product. |
| Annotation\_summary | For each SNP shows the alternative SNPs, e.g. G\_A, the alternative codons, e.g. ACA\_GCA; the alternative amino acids, e.g. T\_A, whether the SNP alleles are Synonymous or Nonsynonymous, and the gene product. If the SNP site is not present in an annotated genome that SNP is shown as Not in annotated genome. If the SNP is in an annotated genome but not in an annotated region it is shown as Not in annotated region of annotated genome. |
| SNPs\_all  SNPs\_in\_majority0.5  core\_SNPs  nonCore\_SNPs | Similar to above, but provides less information: SNP number, context (the SNP allele with the central base indicated by a dot), SNP base, *position in genome, strand,* and genome name. **Italicized information is provided only for genomes listed in the annotate\_list file (argument -annotate)!** |

These files in Table 3 provide a wealth of information about each SNP, some of which is essential if you are exploring the roles of SNPs. For instance, if you have identified certain SNPs as being associated with a particular phenotype, then these files tell you not only where those SNPs are, but what proteins they are in etc.

Table 4. Tree files

|  |  |
| --- | --- |
| Tree files are written for each phylogenetic method that is used. Below each file is described with the word 'method' substituted for the particular phylogenetic method. In each case an example tree using the parsimony method is provided. Methods are parsimony, ML, NJ and majority#.# where #.# can range from 0.0 to 1.0. | |
| Parsimony trees are consensus trees based on an Extended Majority Rule consensus of the equally most parsimonious trees from a sample of 100 trees. | |
| **File** | **Tree** |
| **The tree.method.tre trees have internal node labels that show the support for that node as calculated by FastTreeMP.** | |
| tree.parsimony.tre | A consensus parsimony tree based on all of the SNPs |
| **The tree\_AlleleCounts.method.tre trees have internal node labels that show the number of SNP alleles that are present in all descendants of that node and nowhere else. Allele counts for branch tips are not shown.** | |
| tree\_AlleleCounts.parsimony.tre | A parsimony tree based on all of the SNPs |
| **The tree\_AlleleCounts.method.NodeLabel.tre trees all have internal node labels that show the node numbers corresponding to node numbers in ClusterInfo files and the number of SNP alleles that are present in all descendants of that node and nowhere else.** | |
| tree\_AlleleCounts.parsimony.NodeLabel.tre | A parsimony tree based on all of the SNPs |
| **The tree\_tipAlleleCounts.method.tre files are similar to tre\_AlleleCounts.x.tre trees except that strain names have been modified to show the strain specific allele counts with an “\_” after the strain name.** | |
| tree\_tipAlleleCounts.parsimony.tre | A parsimony tree based on all of the SNPs |

The Neighbor Joining (NJ) trees are based on the number of SNP allele differences between sequences. The distances used for those differences are 2 for locus presence/absence (present in one sequence and absent in the other), 1 for allele differences (both sequences contain the locus but have a different allele), and 0 if the locus is absent in both sequences or both share the same allele.

The 20 files that have the extension .tre are all phylogenetic tree description files in the Newick format. Those files are understood by most tree drawing programs, but we suggest ***Dendroscope*** <http://ab.inf.uni-tuebingen.de/software/dendroscope/welcome.html>, or ***FigTree*** <http://tree.bio.ed.ac.uk/software/figtree/> both of which are available for Mac OS X and Linux platforms, to visualize the trees. The tree\_tipAlleleCounts.x.tre allows you to see the genome-specific allele counts at branch tips in *Dendroscope*, while the tree\_AlleleCounts.x.tre work better for *FigTree*, where allele counts at the leaves can be shown using the tip\_SNP\_counts.x files described below.

It is important to bear in mind that all of the trees are ***unrooted*** trees. The tree drawing programs will, by default, draw them in the rectangular phylogram or cladogram format so that they appear to be rooted. That appearance is for convenience only. There is no evolutionary direction in these trees and you should not infer such a direction. If you have knowledge that a cluster is a true outgroup with respect to the rest of the sequences then you can root the tree on that outgroup in the tree drawing programs, after which you can legitimately infer evolutionary direction.

It is also important to bear in mind that branch lengths are expressed in terms of changes per number of SNPs, not changes per site. When trees are based on comparisons of aligned sequences, branch lengths are expressed in terms of changes per site, which means changes divided by the length of the alignment; many of the sites may be invariant. When trees are based on SNPs there are no invariant sites, so the number of sites is likely to be much smaller. The result is that the absolute value of branch lengths are likely to be much higher than one would see on an alignment-based tree. The relative branch lengths, however, should be very similar. SNP-based branch lengths should not be used for estimating divergence times, but for other purposes should present no problems.

Table 5. Other Output Files

|  |  |
| --- | --- |
| **Various counts of the number of SNPs** | |
| **File name** | **Explanation** |
| COUNT\_SNPs | Shows the total number of SNPs |
| COUNT\_coreSNPs | Shows the number of core SNPs, the number of non-core SNPs, and the number of SNPs in at least  "minimum fraction with locus"  sequences |
| Protein\_Annotation\_counts | Shows the number of SNPs, non-synonymous SNPs, Synonymous SNPs and the NS/S ratio of the SNPs that were annotated, summarized by protein |
|  | |
| **As in Table 4 many kinds of files are written for each phylogenetic method and/or for subsets of SNPs. In each case the word 'method' is substituted for the method or SNP subset. Methods are parsimony, ML, NJ, core, and majority#.#. An example using the parsimony method is provided** | |
| **The COUNT\_Homoplastic\_SNPs files all show the number of homoplastic SNPs (that do not correspond to a node) on the indicated tree.** COUNT\_Homoplastic\_SNPs.method | |
| COUNT\_Homoplastic\_SNPs.parsimony | From the consensus parsimony tree based on all of the SNPs |
|  | |
| **The tip\_SNP\_counts.method files give the number of genome-specific alleles in each genome. Useful to show the strain specific allele counts with FigTree, for the tree\_AlleleCounts trees.** tip\_SNP\_counts.method | |
| tip\_SNP\_counts.parsimony | From the parsimony tree based on all of the SNPs |
|  | |
| **Node\_SNP\_counts** | |
| **The Node\_SNP\_counts files give the number of SNPs that correspond to nodes on the indicated tree, and the genomes under that node.** Node\_SNP\_counts.method | |
| Node\_SNP\_counts.parsimony | From the parsimony tree based on all of the SNPs |
|  | |
| **Homoplasy groups** | |
| **The Homoplasy files give the number of SNPs in each homoplastic group of genomes. These files are similar to the Node\_SNP\_counts files, except for the groups of genomes that share SNPs but that do not correspond nodes of the tree.** They give a group identifier (e.g. “Group.25”  which corresponds to the info reported in the ClusterInfo files), and the number of target sequences that make up this group and the number of SNP alleles that are shared by this group of genomes, followed by the genome identities that make up the group. Homoplasy\_groups.method | |
| Homoplasy\_groups.parsimony | From the consensus parsimony tree based on all of the SNPs |
|  | |
| **ClusterInfo files** | |
| **The ClusterInfo files list for each locus which node or homoplastic group of sequences the locus is present in. Group numbers correspond to the groups listed in the Homoplasy\_groups files.** | |
| ClusterInfo.parsimony | From the parsimony tree based on all of the SNPs |
|  | |
| **Still more files** | |
| **File name** | **Explanation** |
| annotate\_list | List of annotated genomes. Empty if no list was input |
| Annotation\_summary | For each SNP shows the alternative SNPs, e.g. G\_A, the alternative codons, e.g. ACA\_GCA; the alternative amino acids, e.g. T\_A, whether the SNP alleles are Synonymous or Nonsynonymous, and the gene product. If the SNP site is not present in an annotated genome that SNP is shown as Not in annotated genome. If the SNP is in an annotated genome but not in an annotated region it is shown as Not in annotated region of annotated genome. |
| fasta\_list | List of genome fasta files. Identical to the input list file |
| genbank\_from\_NCBI.gbk | The set of GenBank files that were downloaded based on the gi numbers of the finished genomes |
| headers.annotate\_list | list of the fasta header lines for the annotated genomes (needed for annotation) |
| NJ.dist.matrix | A pairwise distance matrix showing the distances between the genomes on the NJ tree |
| unresolved\_clusters | Lists the genomes that fall into unresolved clusters and those that are uniquely resolved. Sequences with the same cluster number in the first column are not resolvable, but those with different cluster numbers are uniquely resolved by SNPs. |
| VCF.reference\_genome.vcf | variant call format file ( <http://en.wikipedia.org/wiki/Variant_Call_Format> ) for compatibility with other tools, using the reference genome indicated in the  file name.   Since some SNPs may not be present in this reference genome, these are listed in the file  VCF.SNPsNotInRef.reference\_genome.  The reference genome is automatically selected to be the first genome in the finished\_genomes file (-annotate option), or if that is empty, the first sequence in the input list (-in option).  Note! These files are generated only if the –vcf option is invoked on the command line.  Be aware that generating these files can use enormous amounts of memory. In a data set consisting of over 200 bacterial genomes and containing over 2,000,000 SNPs generating these files used over 37 GB of RAM. |

.

# X. The kSNP4 Utilities

In addition to the executables (programs) that make up ***kSNP4*** itself there are several "utility" programs that are designed to make your life easier. You have already encountered three utility programs: ***MakeKSNP4infile, genomeNames4,***  and  ***Kchooser4***. This section discusses several other utility programs in detail. The programs fall into three basic categories: (1) programs to help you download genomes from NCBI's ftp archives, (2) programs that you run *before* you run kSNP3, and (3) programs that you run *after* your run kSNP3. You won't always need the utility programs, they are just there to make life easier for you when you do need them.

## Utilities in alphabetical order

### checkGenbankFromNCBI

Problem: during annotation ***kSNP4*** fails with an error message that includes "ParAnn returned -1".

During the annotation process ***kSNP4*** downloads and concatenates all of the Genbank files of the genomes that are specified for annotations. Those Genbank file are concatenated to create the file genbank\_from\_NCBI.gbk. Occasionally one or more files will fail to download and ***kSNP4*** will crash during the annotation process with an error message that includes "ParAnn returned -1".

The utility checkGenbankFromNCBI checks the genbank\_from\_NCBI file by comparing it with the headers.annotate\_list file that lists the GenBank files that *should* have been downloaded. Navigate to the directory that contains the ***kSNP4*** output files. Enter checkGenbankFromNCBI. The accession numbers of the missing GenBank files will be written to a file named missing\_accession\_numbers.txt.

Using those accession numbers and your favorite browser retrieve each of those files from NCBI (<https://www.ncbi.nlm.nih.gov/nucleotide>) and copy the contents of that file starting with the word "Locus" down through the terminator "//" and paste it at the end of the genbank\_from\_NCBI file and save the file.

Finally, navigate to the directory that contains the ***kSNP4*** output files, including the modified genbank\_from\_NCBI file. Enter ParAnn to restart the annotation process and complete the ***kSNP4*** analysis.

### extractNthLocus4

This utility extracts locus number n from a SNPs file. In the outfile it lists, for each genome in which the SNP is present: the SNP number, the sequence surrounding the central base, the position of the SNP in that genome, the amino acid, the genome name, and the genome accesion number.

Example for SNP 9

9 AAACAA.AGGAGC A 1842 F TTV\_T3PB AF247138.1

9 AAACAA.AGGAGC C 1865 F TTV\_BDH1 AF116842.1

9 AAACAA.AGGAGC C 1865 F TTV\_1a AB017610.1

The command is:

extractNthLocus4 n (SNPs\_all or core\_SNPs or SNPs\_in\_majority0.5) > outfileName

Examples:

extractNthLocus4 5 core\_SNPs > locus\_5\_in\_core\_SNPs

extractNthLocus4 155 SNPs\_all > locus\_155\_in\_SNPs\_all

extractNthLocus4 30 SNPs\_in\_majority0.5  > locus\_30\_in\_SNPs\_in\_majority

### fixOldFastaHeaders

Problem: one or more of the genome files includes an old-stlye fasta header that starts with gi|.

The presence of even one such header will cause ***kSNP4*** to crash. It is easy to miss seeing an old-style header for a plasmid or other replicon that is not the chromosome.

Navigate to the directory that *encloses* the directory that contains the fasta genome files and enter:

fixOldFastaHeaders targetFolderName

All old-style headers starting with gi| will be corrected. New-style headers will be unchanged.

### genomeNames4

Extracts the genome names from a ***kSNP4*** input file such as Example1.in and writes a file that lists the genome names.

Usage: genomeNames4 inFile.in annotatedGenomes

### Kchooser4

***Kchooser4*** is discussed extensively in Section VI. The command line is   
Kchooser4 infileName.in where infileName.in is the input file for both ***kSNP4*** and ***Kchooser4***.

### MakeKSNP4infile

***MakeKSNP4infile*** is discussed extensively in Section V.

### NodeChiSquare2tree4

***NodeChiSquare2tree4*** identifies for each node the SNPs for which the 2 probability of the alleles being randomly distributed with respect to that node are ≤ some threshold *p* value, then writes a tree file with the labeled nodes; e.g. tree\_ChiSqAlleleCounts.X.tre where X is the tree type (ML, parsimony, core or majority0.5)

***NodeChiSquare2tree4*** is run from within the ***kSNP4*** output file directory after ***kSNP4*** has run.

Usage: NodeChiSquare2tree4 [-p maximumChiSqProbability -t treeType]

-p maximum Chi Sq probability for assigning a SNP to a node, e.g. 0.05, 0.01

-t tree type from ***kSNP4*** output, possible values are ML, parsimony, core, or 'majority0.5'

-p and -t are optional. Default is -p 0.0001 -t parsimony

It produces two files, NodeChiSquares.txt listing the number of alleles significantly associated with each node, and tree\_ChiSqAlleleCounts.X with a Newick tree showing the number of significant alleles at the nodes.

### rmNodeNamesFromTree4

***rmNodeNamesFromTree4*** removes the labels from internal nodes.

kSNP4 labels the internal nodes with a name. Those labels may interfere with other labels that the drawing program writes. rmNodeNamesFromTree4 simply removes the ***kSNP4*** node labels

Usage:

rmNodeNamesFromTree tree.withNodeLabel.tre tree.nodeLabelsRemoved.tre

The first argument is the name of the tree. The second argument is whatever you want to call that tree with the internal node labels removed

### selectNodeAnnotations4

This utility finds the SNP loci that map to a particular node on a tree.

Usage: select\_node\_annotations3 nodeNumber treeMethod

Begin by displaying the AlleleCounts - NodeLabel tree in DendroScope or FigTree, e.g. tree\_AlleleCounts.ML.NodeLabel.tre. In FigTree be sure that it is set to show node labels. Each internal node is labeled with the node number and the allele count, separated by an underscore; e.g. 9\_227 which means node 9, 227 SNPs.

To find all the SNPs that map to node 9 on the ML tree you would enter selectNodeAnnotations4 9 ML.

The NodeNumber can either be the number of an internal node or the name of a tip node.

Two output files are written: (where X stands for the node number)

1. node.X.treeMethod.annotations, which gives the annotation(s) for each SNP that is specific to that node
2. node.X.treeMethod.loci, which lists the IDs of the SNPs that are specific to that node

If you just enter selectNodeAnnotations4 without any arguments instructions will be displayed on the screen.

### fixOldFastaHeaders

Problem: one or more of the genome files includes an old-stlye fasta header that starts with gi|.

The presence of even one such header will cause ***kSNP4*** to crash. It is easy to miss seeing an old-style header for a plasmid or other replicon that is not the chromosome.

Navigate to the directory that *encloses* the directory that contains the fasta genome files and enter:

fixOldFastaHeaders targetFolderName

All old-style headers starting with gi| will be corrected. New-style headers will be unchanged.

# XI How kSNP4 works

The ***kSNP4*** software finds single nucleotide polymorphisms (SNPs) in whole genome data. SNP discovery is based on k-mer analysis, and requires no multiple sequence alignment or the selection of a single reference genome. A SNP locus is defined by the k-mer sequence surrounding the central base, which is the SNP allele, where a k-mer is an oligo of length k.

The process is outlined in Figure 1. kSNP3 enumerates the all the k-mers in each genome using jellyfish (step 1). It calculates the average of the mean and median number of kmers from the kmer frequency distribution for each genome, and eliminates kmers occurring less than this number. This is a heuristic that allows a flexible kmer count threshold for each genome that depends on the coverage for unassembled genomes, and always results in a threshold of 1 for assembled genomes. (step 2). It removes k-mers within each genome that would result in allele conflicts, that is, two or more k-mers in a single genome that only differ in the center base, since each genome can only have a single allele at a given locus (step 3). Then it compares the k-mers across genomes to find SNP loci, that is, k-mers in which there are allele differences among at least 2 genomes (steps 4-5). It reports the SNP allele in each genome by comparing the k-mer list for that genome with the SNP loci (step 6). In genomes for which the user has specified that it should find SNP positions, it finds the position and strand using MUMmer (step 7). SNP matrices are created from all the SNPs, only the core SNPs, or SNPs that occur in at least the user specified number of genomes, and trees are built using parsimony, neighbor joining, and maximum likelihood (step 8). Tree nodes are labeled with the number of SNPs that correspond to each node or leaf, and SNPs that do not correspond to a branch of a tree are grouped into sets of genomes that share an allele (step 9). Finally, SNPs are annotated with any information available in Genbank or in a GenBank formatted file uploaded by the user for information about, for example, whether a SNP occurs on a protein CDS or mature peptide, and causes an amino acid difference, or whether it occurs on a 3’ UTR, etc. (step 10).

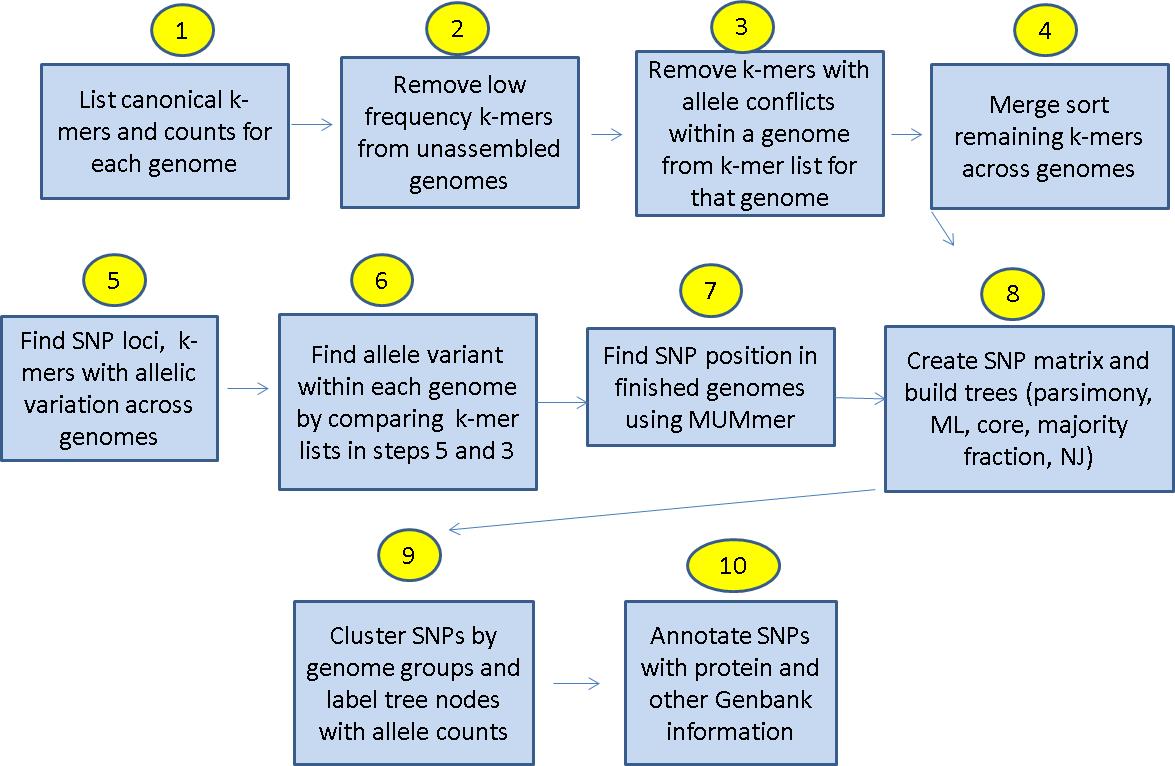


Figure 1: diagram outlining kSNP3 process

# XII Citations

If you use ***kSNP4*** please cite

**kSNP3:** Gardner, S.N., T. Slezak, and B.G. Hall. 2015. kSNP3.0: SNP detection and phylogenetic analysis of genomes without genome alignment or reference genomes. Bioinformatics **31:** 2877-2878 doi: 10.1093/bioinformatics/btv271.

**kSNP v2:** Gardner, S. N. and B. G. Hall. 2013 When whole-genome alignments just won't work: kSNP3 v2 software for alignment-free SNP discovery and phylogenetics of hundreds of microbial genomes.

PLoS One 8(12): e81760. doi:10.1371/journal.pone.0081760

Citation for kSNP v1 is:

Gardner SN, Slezak T. Scalable SNP analyses of 100+ bacterial or viral genomes. 2010. J. Forensic Research, 1:107.

***kSNP4*** incorporates six third-party programs, which we gratefully acknowledge. These are:

**Jellyfish:** Guillaume Marcais and Carl Kingsford, A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics (2011) 27(6): 764-770 (first published online January 7, 2011) doi:10.1093/bioinformatics/btr011

**FastTree:** Price, M.N., Dehal, P.S., and Arkin, A.P. (2010) FastTree 2 -- Approximately Maximum-Likelihood Trees for Large Alignments. PLoS ONE, 5(3):e9490. doi:10.1371/journal.pone.0009490.

**Parsimonator:** A. Stamatakis distributed under GNU GPL via [www.exelixis-.lab.org](http://www.exelixis-.lab.org/) and <https://github.com/stamatak>.

**Mummer:** Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg (2004) Versatile and open software for comparing large genomes. Genome Biology **5**: R12

**sa:** Hysom DA, Naraghi-Arani P, Elsheikh M, Carrillo AC, Williams PL, et al. (2012) Skip the Alignment: Degenerate, Multiplex Primer and Probe Design Using K-mer Matching Instead of Alignments. PLoS ONE 7(4): e34560. doi:10.1371/journal.pone.0034560

**Consense**: Consense is part of the Phylip suite of programs distributed via <http://evolution.genetics.washington.edu/phylip.html> Felsenstein, J. 2005. PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

# XIII Licenses

***KSNP4*** by Barry G. Hall, Jeremiah Nisbet and Shea N. Gardner is distributed under the terms of the BSD OPENSOURCE LICENSE.

***kSNP4*** is derived from and based upon ***kSNP3*** by Shea N. Gardner and Barry G. Hall

Copyright (c) 2014, Lawrence Livermore National Security, LLC.

Produced at the Lawrence Livermore National Laboratory

Written by Shea N. Gardner, gardner26@llnl.gov

kSNP3 is LLNL-CODE-610052

kSNP3

by Shea N. Gardner

kSNP3 OCEC-12-091

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer (as noted below) in the documentation and/or other materials provided with the distribution.

Neither the name of the LLNS/LLNL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL LAWRENCE LIVERMORE NATIONAL SECURITY, LLC, THE U.S. DEPARTMENT OF ENERGY OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING

IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Additional BSD Notice

1. This notice is required to be provided under our contract with the U.S. Department of Energy (DOE). This work was produced at Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344 with the DOE.

2. Neither the United States Government nor Lawrence Livermore National Security, LLC nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights.

3. Also, reference herein to any specific commercial products, process, or services by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Jellyfish is distributed under the terms of the GNU GENERAL PUBLIC LICENSE. Source code for Jellyfish is freely obtainable at <http://www.cbcb.umd.edu/software/jellyfish/>.

FastTreeMP is distributed under the terms of the GNU GENERAL PUBLIC LICENSE. Source code for FastTreeMP is freely obtainable at [http://www.microbesonline.org/fasttree/ - Install](http://www.microbesonline.org/fasttree/#Install).

Parsimonator is distributed under the terms of the GNU GENERAL PUBLIC LICENSE. Source code for Parsimonator is freely obtainable at <http://www.exelixis-lab.org/> by going to the Software tab and scrolling down to Parsimonator.

Mummer is distributed under the terms of the Open Source Initiative License. Source code for Mummer is freely obtainiable at <http://mummer.sourceforge.net/>

sa is part of PriMux which is available as open source on sourceforge <http://sourceforge.net/projects/primux/> .

Consense, part of the Phylip suite, is distributed under an open-source license. See <http://evolution.genetics.washington.edu/phylip/doc/main.html>.

# XIII FAQ (Frequently Asked Questions)

Who should I contact for support of kSNP3?

Unfortunately there is no "support team" for ***kSNP4*** and we can offer only the most limited support.  Before asking for support please answer the following questions:

1. Are you using the most recent version of ***kSNP4***?  If not, please download and install the most recent version which is currently v4.0 and try again.
2. Did you download one of the packages of executables (Linux or Mac), or did you download the source code?  If you are trying to run ***kSNP4*** from the source code we can offer no support whatsoever.  Download one of the executable packages and try that.
3. Did you install the ***kSNP4*** executables EXACTLY as described in the User Guide?  If not, do so and see if your problem disappears.

If you are still unable to resolve your problem, or if you are confident that your problem comes from a bug, please direct enquiries to barryghall@gmail.com. Because of our many other commitments you should not expect a prompt response to such enquiries.

How can I easily download the genomes I want to analyze with ***kSNP4***?

Read "Downloading Genomes from NCBI.pdf" in the documentation directory.

Should I download the source code or one of the packages of executables?

We strongly encourage users to download one of the packages of executables and to install that package exactly as described in this user guide. Note that we offer no support whatsoever for kSNP3 installations of the source code.

Do I have to uninstall ***kSNP v3*** from my computer in order to run ***kSNP4***?

No, you can keep both versions if you want to. The earlier version will not interfere with kSNP4.

How do I uninstall ***kSNP3*** or ***kSNP4***from my computer?

Simply delete the ***kSNP3*** or ***kSNP4*** directory from /usr/local. After that remove kSNP3 from your path variable.

How do I find which SNP loci map to a particular node on a tree?

We have provided a little utility as a bash shell script for exactly this purpose. The utility is named **selectNodeAnnotations**. See Section X, the kSNP4 Utilities section under Utilities to run after running kSNP3. Discard this or rewrite as bash script?

How much time is required for a kSNP3 run?

That depends on the size of the data set, i.e. the number of genomes and the sizes of those genomes; on the number of finished genomes that will be used for annotations, and on the number of SNPs (which you cannot know in advance) and on your computer's CPU.

How much RAM does ***kSNP4*** require?

That is a function of the data set. We recommend at least 16 GB of RAM, the more the better.

How can I extract the nth locus from the core\_SNPs file?

The tcsh script extractNthLocus does exactly what you want. See section X, the kSNP3 Utilities section.

All of the trees have some sort of label at the internal nodes. How can I remove those labels for the purpose of drawing the tree?

The utility rmNodeNamesFromTree does exactly what you want. See Section X the kSNP4 Utilities.

Are line endings an issue for the genome fasta files?

Computer text files are cursed with three alternative characters that indicate the ends of lines: The Unix/Linux/Mac OSX platform recognizes the LF character; the old classic Mac platform uses the CR character; and the Windows/DOS platform recognizes CRLF (CR followed by LF) as ending a line. kSNP4 expects all input files to have Unix line endings.

If you downloaded the genome files from NCBI they should have Unix line endings. If you were given genome files by a colleague some of the files may have Windows or (probably rarely) classic Mac line endings. Those files will not work with kSNP4. If you are in doubt look at the file in your text editor and note which line endings are present. To avoid that problem both the latest iteration of kSNP4 (and later) check each genome file for its line ending character. If the file has Windows line endings the programs rewrite the files to have Unix line endings, a process that is transparent to the user. If kSNP4 finds classic Mac line endings it reports them on the screen and also warns the user to change the line endings manually. However, it then continues one with the rest of the program so in most cases the warning will not be noticed. kSNP4 winds up simply ignoring those genome sequences, so they are not included in any of the output including tree files. So, if you notice that some genomes are missing from the output look near the beginning of the Log file (you did remember to use | tee logfileName on the command line so that the logfile duplicates all screen output, didn't you???).

It is **essential** that fasta genome files of the genomes in that list have the new-style fasta headers. The old-style header line begins with gi and looks like this:

>gi|49482253|ref|NC\_002952.2| Staphylococcus aureus subsp. aureus MRSA252 chromosome, complete genome

In 2016 NCBI discontinued using gi numbers.

The new style header line begins with an accession number and looks like this:

>NC\_002952.2 Staphylococcus aureus subsp. aureus strain MRSA252, complete genome

Any files in the list of annotated genomes that have the old-style headers must be fixed by running the program ***fixOldFastaHeaders s***. See Section X the kSNP4 Utilities.

Can I edit the genome IDs in an input file written by MakeKSNP4infile in the automatic mode?

Absolutely. You can edit the genome IDs however you like, and if you have moved a file you can edit the path name accordingly.

If I didn't originally run kSNP4 with the -annotate option, can I do annotation after the fact?

No. You need to repeat the run with the -annotate option.

Can I use raw-read files in the fastq format for kSNP4?

The short answer is, no you cannot. All input files must be in fasta format.

You need to convert the fastq to fasta, using a tool that will convert low quality bases to N. It is really important to turn low-quality bases into N, otherwise the reliability of those raw-read sequences will be too low to trust at all.

A Google search reveals that there are numerous conversion programs available, but most of them require pretty sophisticated computer/programming skills to install and use.

As it turns out that conversion is not at all a trivial matter.  Each sequence in a fastq file consists of a header line, a line consisting of the sequence itself, an essentially blank line, and a quality line.  The quality line has, for each base on the sequence line, a corresponding character that indicates the quality of that base - essentially the probability that the base has been correctly called.  If there were a single standard for what the characters on the quality line mean it would be easy to translate fastq to fasta, substituting N for low-quality bases.     Sadly, there is no such single standard.  Not only are there different ranges for the quality score itself, the characters corresponding to those scores differ.  Solexa/Ilumina 1.0 uses one coding, Ilumina 1.3 uses a different coding, Ilumina 1.5 still a different coding. See <https://en.wikipedia.org/wiki/FASTQ_format> for a discussion of fastq and quality scores.  In practice, that means that you need to know what manufacturer and what version of the software generated the fastq file in order to convert it to fasta while safely converting low-quality bases to N.  You will need to consult with the proper manufacturer to solve that problem.

If you have other questions or find bugs, email Barry G. Hall: barryghall@gmail.com.

# Appendix I Suggested text editors

**•** For Macintosh we recommend the free TextWrangler from BareBones Software (<http://www.barebones.com/>) or the more sophisticated commercial BBEdit from the same source.

• For some Linux OS, those that use the Gnome desktop GUI, gedit is the default text editor (although it might just be called "Text Editor" in the sidebar). For other versions of Linux download Gedit from <https://wiki.gnome.org/Apps/Gedit>.

In both cases the Text Editor should be configured to save files with **Unix** line endings, and it is helpful to configure it to show line numbers and not to wrap text.

# Appendix II Details of Downloading genome sequences and preparing an input file.

It is convenient to store the finished genomes and genome assembly files in different directories, so I will illustrate a two-step downloading process.

**Step 1: Get the finished genomes**

Create a directory for the species, i.e. Vibrio\_cholerae. From within that directory run the utility FetchFinishedGenomes as described in the Utilities to Download genomes section. A directory named GenomeFiles will be created. I suggest renaming it to include the name of the organism; i.e. VibrioFinishedGenomes or something like that.

Within that folder there will be a separate directory for each genome, plus a file entitled FinishedGenomeNames. Each genome directory will contain a .fna file for each replicon in the genome, plus a .fasta file that includes all of the replicons. The fasta file will have a name something like Vibrio\_cholerae\_IEC224.fasta. That genome ID is what will appear in all of the kSNP output files, including the tree files. It may be convenient to shorten the name to something like Vc\_IEC224.fasta. If you do that be sure to change the names in the FinishedGenomeNames file as well

Make a new directory within the GenomeFiles directory named FinishedFastaGenomes and put each of the .fasta genome files into that directory.

**Step 2: Get the genome assemblies**

Run the utility FetchGenomeAssemblies as described in the Utilities to Download genomes section. A directory names Assemblies will be created. I suggest renaming it to include the name of the organism; i.e. VibrioAssemblies or something like that.

Within that folder there will be a single .fasta file for each genome, and that file will include all of the contigs in the assembly. Again, it may be convenient to shorten the names to something like Vc\_42608.fasta.

**Step 3: Put the species directory wherever you want it and leave it there.**

**Step 4: Make the kSNP input list file from the finished genomes**

Run the utility MakeKSNP3infile as described in the section Utilities to use before running kSNP3. Navigate to the finished genomes directory and enter MakeKSNP3infile FinishedFastaGenomes finished\_inlist A. This will generate a file named finished\_inlist.

**Step 5: Make a kSNP3 input list file from the genome assembly files**

Navigate to the Assemblies directory and enter MakeKSNP3infile AssemblyFiles assemblies\_inlist A. This will generate a file named assemblies\_inlist.

**Step 6: Combine the two inlist files**

Open finished\_inlist, then open assemblies\_inlist, copy the entire file, and paste it at the end of the finished\_inlist file and save the file as something like Vibrio\_inlist.

**Step 7. Move the final inlist file and the FinishedGenomeNames file to a directory from which you will run kSNP3.**

The inlist file is the main input (-in) for kSNP3, and the FinishedGenomeNames file will be used as the argument to the -annotate option if you want to annotate SNPS during the kSNP3 run.