November 5, 2019 Current version: 3.1.2

kSNP3 Users Guide

Shea N. Gardner, Lawrence Livermore National Laboratory
and
Barry G. Hall, Bellingham Research Institute

kSNP3 Programming by Shea N. Gardner and Barry G. Hall
Utilities Programming by Barry G. Hall and Shea N. Gardner
Documentation by Barry G. Hall and Shea N. Gardner

Download either the **kSNP3 Mac Package** or the **kSNP3 Linux Package**. Each package contains a **kSNP3** directory and this documentation. Also download the two example directories, each of which contains input files for the examples discussed in section IV. Installation of kSNP3 is discussed in section VII.

Table of Contents

LLNL document release number: LLNL-CODE-666381

Preface: a note to users of earlier versions

In 2016 NCBI discontinued the use of gi numbers as identifiers for sequence files. At that point the SNP annotation feature of kSNP3 stopped working because that feature depended on gi numbers. kSNP v3.1 restored the annotation function by making it independent of the gi function. However, the fasta genome files of the reference genomes that are used for annotation purposes ***must*** have the new-style fasta headers in order for SNP annotation to work. 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
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

If you already have fasta files of a lot of finished genomes on your computer it is not necessary to replace them. It ***is*** necessary to replace the fasta headers in any that you intend to use as references for annotation purposes. The utility program fix_old_fasta_headers replaces the any old style headers with new style headers, and it is a good idea to run that program on any genomes that you might use for annotation purposes. See the section on Troubleshooting kSNP3 for specific directions.

The –all_annotations command of version 3.021 no longer has any effect.

The files in the old Examples folder had old-style fasta headers and will therefore not work for annotations. Be sure to download the current Examples folder from SourceForge.

I. The kSNP3 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.

kSNP3 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, kSNP3 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. kSNP3 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 kSNP3 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. kSNP3 also provides a separate analysis of those SNPS that occur in at least a user-determined fraction of the genomes.

kSNP3 is provided as both a Linux package for 64-bit CPUs and as a Mac OS X package. The packages contain the kSNP3 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.

kSNP3 incorporates six third-party programs. Please see section VI, Citations, for the list of publications describing these software packages.

II. kSNP3 vs kSNP v2: What's new and why switch to kSNP3?

kSNP3 differs from kSNP v2 in several ways:

1. kSNP3 allows for multiple replicons in a genome; e.g. two chromosomes as in *Vibrio cholera,* or genomes that include one or more plasmids. Now, when multiple replicons are present SNPs are correctly annotated with respect to each replicon. This was not possible in kSNPv2.
2. The input file formats are completely different.
3. The command line arguments are completely different.
4. The kSNP3 parsimony tree is a consensus of up to 100 equally parsimonious trees.
5. The files that are written by default are different.
6. kSNP3 is faster (see Section IX Time and Efficiency Considerations).
7. There is now an option to add genomes to an existing SNP run instead of doing SNP discovery. It will search the added genomes for the SNPs already found in a previous kSNP3 run.
8. The code automatically determines which genomes are high coverage raw reads versus those that are either assembled or low coverage, and automatically picks the minimum kmer frequency for consideration as a SNP as a proxy for coverage.

kSNP v2 users are **strongly** encouraged to examine Table 1 (command line arguments) and to read the section on input files carefully.

In addition, several utility programs, designed to make life easier for kSNP3 users, have been added. kSNP v2 users should look through that list of utility programs

III. Running kSNP3

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".

kSNP3 is run from the command line within the ***Terminal*** program. ***Terminal*** is found within the Accessories directory (Linux) or Utilities directory (Mac) under Applications. In order for ***Terminal*** to interact with input files it is necessary to set the Run directory (the directory containing the input files) as the current directory. To do that type `cd` , then drag the Run directory into the ***Terminal*** window and hit the Enter or Return key. (Don't forget the space after `cd`). In general, setting a directory as the current directory is known as "navigating to the directory". In ***Terminal*** all commands are completed by hitting the Enter key. In the instructions below what you type and what is printed to the Terminal screen is shown in `Courier` font.

In ***Terminal*** enter the word `kSNP3` followed by a series of *arguments*. (From now on we won't say "in ***Terminal*** ", the instruction to enter something will always mean "in ***Terminal*** "). Arguments are instructions that tell kSNP3 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. 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 in_list`	required
directory for output files	-outdir	name of directory	`-outdir Run1`	required
k-mer size	-k	odd integer	`-k 13`	Required, length of sequence flanking the SNP
file of genome names to use as references for gene annotation of SNPs	-annotate	file name	`-annotate annotated_genomes`	optional
Path to SNPs all file from a prior run	-SNPs_all	full path to SNPs_all	`-SNPs_all /Users/barry/desktop/SNPs_all`	optional Add genomes to a prior analysis, does not find new loci, but does find known loci in new genomes.
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	`-CPU 12`	optional, defaults to the number of processors available

		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 the bottom of this page for the rules about naming files.

Example: `kSNP3 -in in_list -k 13 -outdir Run1 -annotate annotated_genomes`

The purposes of the optional arguments are discussed below.

Capture a Logfile

kSNP3 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. There are two ways you can capture the screen output as a log file:

1. Use "tee" to send the output to both the screen and a file. In the bash shell (default on the Mac and many Linux versions) the command is implemented by adding `| tee Logfile.txt` after the usual kSNP3 commands, for example

```
kSNP3 -in in_list -k 13 -outdir Run1 -annotate annotated_genomes | tee Run1Logfile
```

   In the tcsh shell it is implemented by adding `| & tee Logfile.txt`, e.g.

```
kSNP3 -in in_list -k 13 -outdir Run1 -annotate annotated_genomes  | & tee Run1Logfile
```

2. If you forget to use the "tee" option you may be able to capture the output after the run is complete by selecting all the output in the Terminal window, copying it, and pasting it into a new file in your favorite text editor. That works well on the Mac, but may not work on all systems.

The input files

All of the input files are simple text files. Word processor files written by Microsoft Word, WordPerfect, etc. will not work.

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.
2. The file name may not contain any spaces. Escherichia_coli_042.fasta is legal; Escherichia coli 042.fasta is not.
3. The file ID portion of the file name cannot be purely numeric. EC9123.fasta is legal; 9123.fasta is not.
4. 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 kSNP3 finds a file with an illegal character it terminates the run and writes a file named NameErrors.txt. NameErrorss.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:

```
Line 2: CFSA 003.fasta
Line 3: CFSA:004.fasta
```

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.

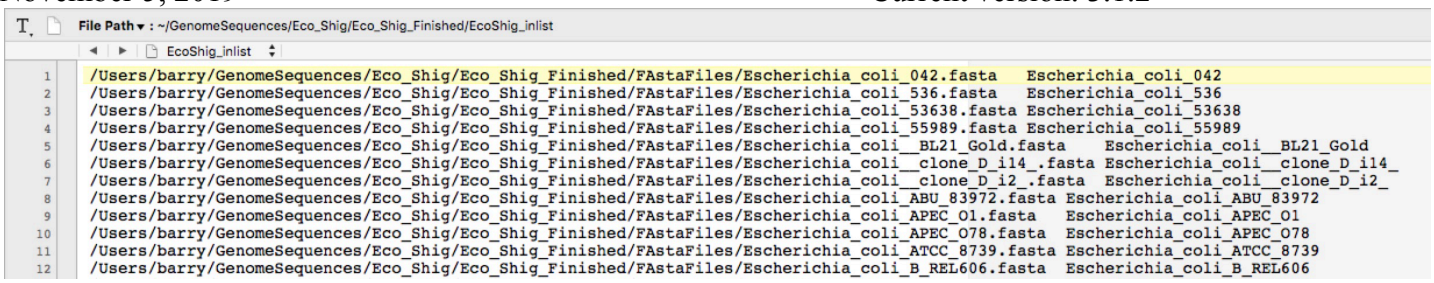
Correct the illegal file names on the files themselves and in the input file!!! Once that is done throw the NameErrors.txt file into the trash. That is essential because the way that kSNP3 knows to terminate the run is by looking for a file named NameErrors.txt. If it finds that file, even if it a left-over file, kSNP3 will terminate the run.

Input list file:

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 below)** The genome can be a finished (closed) sequence, multiple chromosomes and plasmids, an assembly of multiple contigs, or raw, unassembled reads. kSNP v2 required that all of those genome sequences be combined into a single fasta file (thus doubling the amount of hard drive space used by the original files), then made *another* copy of the file in the specific fasta format required by kSNP v2; now tripling the drive space that was used. Aside from the space used, those files were often so large that they could not be edited by text editing programs.

kSNP3 takes an entirely different approach. The input file is not the sequences themselves, but 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. This means that the fasta files each contain only a single genome so are smaller, making them easier to open, edit etc. It also means that you do not need a copy of the genome sequence for every run of kSNP3 on a different input set containing that genome. The input list simply tells kSNP3 where to find each genome sequence file.

The input file might look like this:


File Path ▾ : ~/GenomeSequences/Eco_Shig/Eco_Shig_Finished/EcoShig_inlist

EcoShig_inlist

```
1  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_042.fasta    Escherichia_coli_042
2  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_536.fasta    Escherichia_coli_536
3  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_53638.fasta Escherichia_coli_53638
4  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_55989.fasta Escherichia_coli_55989
5  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli__BL21_Gold.fasta    Escherichia_coli__BL21_Gold
6  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli__clone_D_i14_.fasta Escherichia_coli__clone_D_i14_
7  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli__clone_D_i2_.fasta  Escherichia_coli__clone_D_i2_
8  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_ABU_83972.fasta Escherichia_coli_ABU_83972
9  /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_APEC_O1.fasta   Escherichia_coli_APEC_O1
10 /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_APEC_O78.fasta  Escherichia_coli_APEC_O78
11 /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_ATCC_8739.fasta Escherichia_coli_ATCC_8739
12 /Users/barry/GenomeSequences/Eco_Shig/Eco_Shig_Finished/FAstaFiles/Escherichia_coli_B_REL606.fasta  Escherichia_coli_B_REL606
```

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 only downside of this approach is that genome sequence files *must not be moved to new locations* or the list will no longer point to the file and kSNP3 won't find it. This requires some organization. It is better to set directories and subdirectories to hold genome sequences that are independent of the current data set of interest, rather than putting the genome sequences into the same directory as the kSNP output.

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 kSNP3 input file will be wrong and kSNP3 won't be able to find the genome files - nothing will work. You would have to reconstruct every affected input file. Awful!

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.

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.

The line endings 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. kSNP3 expects all input files to have Unix line endings.

Depending on their origins and how they may have been downloaded some of the genomes sequence files may have Windows or (probably rarely) classic Mac line endings. Those files will not work either with kSNP3 or with the MakeFasta utility program. To avoid that problem both the latest iteration of kSNP3 (v3.1 and later) and MakeFasta 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 MakeFasta finds classic Mac line endings it aborts, lists all of the genome files with classic Mac line endings, and directs the user to change those line endings manually. If kSNP3 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. kSNP3 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???).

Three ways to make an input list file

Manually: as described above, 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.

Semi-automatically using the utility program ***MakeKSNP3infile*** (See section VI, The kSNP3 Utilities, under Utilities to use before running kSNP3).
Gather all of the genome files into a single directory (folder) and *put that directory where you intend to keep it!* If you move the directory with the genome files then the paths in the kSNP3 input file will no longer be correct.

In terminal navigate to that directory and enter `MakeKSNP3infile myInfile S`. 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.

Fully automatically using the utility program ***MakeKSNP3infile*** (See section VI, The kSNP3 Utilities, under Utilities to use before running kSNP3).
Gather all of the genome files into a single directory and *put that directory where you intend to keep it!* If you move the directory with the genome files then the paths in the kSNP3 input file will no longer be correct.

Note: The automatic mode is useful only when the fasta file name is the genomeID that you want.

In terminal navigate to that directory and enter `MakeKSNP3infile myInfile A`. myInfile is the name for the kSNP3 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).

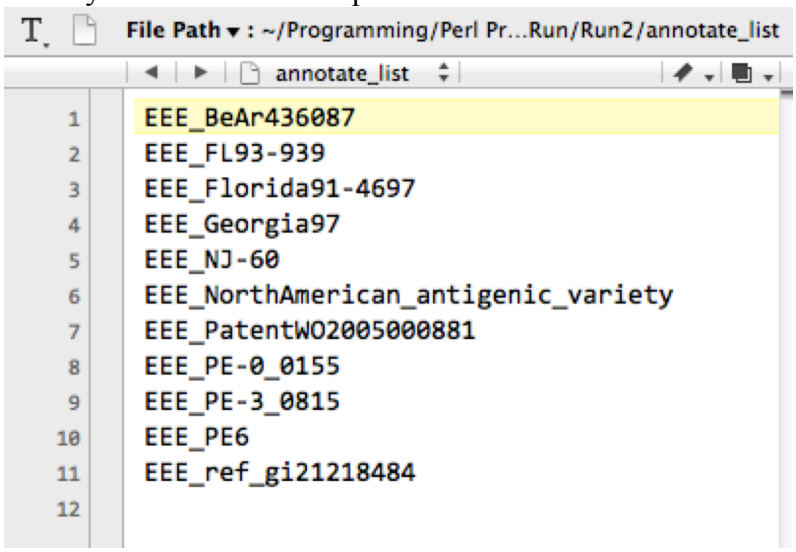
Converting kSNP v2 fasta input files to individual files for input to kSNP3

The program ***FG2IF*** (fasta genomes to individual files) takes a fasta file of many genome sequences and converts them to a set of individual fasta files that can then be used to generate the input file for kSNP3. See ***FG2IF*** in section VI, The kSNP3 Utilities, under Utilities to use before running kSNP3. You can then run `MakeKSNP3infile`.
Example of how to create kSNP3 input from kSNP v2 input:

1. To put a separate fasta file for each genome into a directory named GenomesDirectory for all the genomes in a file named fastainput_fromkSNPv2 enter:
   ```
   FG2IF  fastainput_fromkSNPv2  GenomesDirectory
   ```
2. Create the file for the kSNP3 –in option:
   ```
   MakeKSNP3infile GenomesDirectory  kSNP3_in_list   A
   ```

File of annotated genomes

The file simply lists the genome *names* (not the paths, and not the file names) of the sequences for which you would like SNP positional information as shown below.



That is usually a list of finished sequences, although the list can also include annotated genome assemblies. ***A file of annotated genome names is required in order for positional information and annotation of SNPs to be reported!*** The program **genome_names3** can be used to extract the names of the annotated genomes from the input list file. One would first enter the annotated genomes into the input list, then in terminal enter: `genome_names3 input_list annotated_list`. 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.

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 ***fix_old_fasta_headers***. (See Utility Programs)

Annotating SNPs

To annotate SNPs use the –annotate option followed by the name of the files of annotated genome names.
Example: `kSNP3 -in in_list -outdir Run1 -k 19 -annotate annotated_list`

Other arguments

kmer size: kmer size, *which must be an odd number*, defines the length of the oligonucleotides (kmers) that kSNP3 identifies in all of the sequences. Oligos that are identical between different genomes at all but the

central base are taken as being homologous SNPs, i.e. a SNP locus. If 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 to 51 bp, then because of frequent sequence variation at multiple sites within the oligo, many SNPs will be missed, since 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. **Kchooser** helps pick an optimum value for K and it is strongly suggested to run Kchooser before running kSNP3. Kchooser is described in detail in Section VI kSNP3 Utilities under Utilities to use before running kSNP3

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.

kSNP3 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.

Issue with building NJ trees with large data sets

We have noticed that when analyzing a large data set (207 bacterial genomes, 1,388,522 SNPs) it took 26 hours to build the distance matrix from which the NJ tree was computed. This was just over half of the total time for the analysis. Our simulation studies also found that NJ trees were not very accurate. For that reason the default is to **not** to calculate an NJ tree. If you want to calculate an NJ tree just add the argument `-NJ` to the command line (Table 1). You will probably find it faster to calculate an NJ tree externally with a phylogenetics program such as MEGA http://www.megasoftware.net/ by using SNPs_all_matrix.fasta, core_SNPs_matrix.fasta, or SNPs_in_majority0.5_matrix.fasta as the input alignment.

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 ***NodeChiSquare2tree3*** 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).

NodeChiSquare2tree3 is run from within the kSNP3 output file directory after kSNP3 has run. Decide if you want to run ***NodeChiSquare2tree3*** after examining the tree_AlleleCounts.X.tre file of interest.

Adding genomes to an existing kSNP3 run

It can be useful to add another genome to a kSNP3 run that you have already done. There is a small section in the tutorial in Section IVB that illustrates adding another genome.

When you add another genome the trees are revised to include the added genome(s), but you should be aware of the limitations of the process. During addition kSNP3 only looks for the old set of SNPs in the new genome, it doesn't look for new SNPs in the added genome. The total number of SNPs remains the same.

Likewise, annotation doesn't work with the genome addition function. The identified SNPs remains the same and the annotations remain the same. kSNP3 does not yet update the annotations files to indicate SNPs that are present in the new genome. Doing so is on the to do list for a future release, but is not part of this release.

If complete annotation and identification of SNPs specific to the new genome are important you will need to re-run kSNP3 with the updated input list.

IV. Tutorials with Examples

Because kSNP3 was designed to maximize flexibility, and in anticipation of future uses of comparative SNP information, kSNP3 writes a plethora of output files. Indeed, analysis of just 11 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 kSNP3, it will allow you to determine whether you have installed kSNP3 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.

IVA 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 the data set within a directory named Genomes. That data set consists of finished sequences of 10 encephalitis virus genomes. It also contains a directory named "ExampleRun" Within that is the same data set, the input files, and the results of Runs 1, 2 & 3 done by the authors. 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 files

1. Generate the input file list by running ***MakeKSNP3infile***. Navigate to the Example1 directory and enter `MakeKSNP3infile Genomes in_list A .` The result will be an input list file named in_list. Generate the list of annotated genomes. Since all of the genomes are annotated you can extract the list from in_list. For the sake of this tutorial call the output file annotated_genomes, but you could call it anything you like. Enter `genome_names3 in_list annotated_genomes.`

Step 2: Run kSNP3

2A **without annotations**

The simplest way to run kSNP3 is without doing any SNP annotation. We will set the kmer size to 13, but see the discussion of Kchooser to learn how that kmer size was chosen. We will call this run Run1. All the options except -in, -outdir, and -k take their default values.

Enter `kSNP3 –in in_list –outdir Run1 –k 13 | tee Run1Log`

(for tcsh systems use `|& 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 0.02 hours, or 72 seconds. Note that in the position column 4 of SNPs_all, 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 `kSNP3 -in in_list -outdir Run2 -k 13 -annotate annotated_genomes | 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 kSNP3 `-in in_list -outdir Run3 -k 13 -annotate annotated_genomes -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.

IVB Tutorial Example2: Finished and unfinished genomes

Before discussing this example, a quick word about unassembled genomes. Before running kSNP3, 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 kSNP3 handles and annotates multi-chromosome genomes, (3) to illustrate the use of Kchooser to determine the optimum kmer size, and (4) to illustrate how to add a genome to a data set that has already been

analyzed by kSNP. You should open one of the chromosome files and look at it to see how such genomes should be made into fasta files.

The Example2 directory contains six items: (1) a directory named Genomes that contains all of the fasta files, (2) a kSNP3 input file named "in_list" that contains paths and genome names for five genomes. ***You cannot use that file for input to your kSNP3 program!!!*** The paths in that file are not correct because the Genomes directory is somewhere on your hard drive and the paths point to the directory on Barry Hall's machine. See the next paragraph for what to do. (3) a file named "in_list2" that will be used in the second stage of this tutorial to add a 6th genome. That file, also cannot be used without modification. (4) a file named "annotated_genomes" that lists the genome IDs of the four annotated genomes. That file can be used as it is because it does not contain any paths. (5) A directory named ExampleRuns that contains the results of runs by the author for comparison with your results, and (6) a file of the command lines for this example.

We will begin by modifying the input list file in_list so that you can use it. Open in_list in a text editor program. It will look like this.

```
/Users/barry/Programming/Perl Programs/Barry's Perl Scripts/kSNPv3/Examples/Example2/Genomes/ERR579925.fasta	ERR579925
/Users/barry/Programming/Perl Programs/Barry's Perl Scripts/kSNPv3/Examples/Example2/Genomes/Vc523-80.fa	Vc523-80
/Users/barry/Programming/Perl Programs/Barry's Perl Scripts/kSNPv3/Examples/Example2/Genomes/VcLMA3984-4.fa	VcLMA3984-4
/Users/barry/Programming/Perl Programs/Barry's Perl Scripts/kSNPv3/Examples/Example2/Genomes/VcMS6.fa	VcMS6
/Users/barry/Programming/Perl Programs/Barry's Perl Scripts/kSNPv3/Examples/Example2/Genomes/VcO1-ElTorN16961.fa	VcO1-ElTorN16961
```

The paths are at the left, separated by tabs from the genome IDs. Replace the paths to point to where you have the files. Mac OSX users can simply delete a path, then Command-drag the corresponding file into the document to replace the incorrect path. Many Linux uses can Control-select the file in the Genomes directory, choose Copy from the drop-down menu, then paste the path into the document. When you have finished modifying in_list so that it will work on your computer, modify in_list2 similarly.

`The` ERR579925.fasta file was created by downloading the genomes from the NCBI SRA database, and filtering/trimming low quality bases and reads.

Before running kSNP3 we need to know the appropriate kmer size, which will be calculated by ***Kchooser***. ***Kchooser*** requires a fasta file for its input so we will use ***MakeFasta*** to create that file. (See section VI, The kSNP3 Utilities, under "Utilities to use before running kSNP3").

Make a copy of in_list and delete from that copy the raw-reads genome ERR579925.fasta.

Enter `MakeFasta in_list_copy VC.fasta`. VC.fasta is the fasta file that will be created (named VC for *vibrio cholera*). After the VC.fasta file appears enter `Kchooser VC.fasta`. After a bit a file named Kchooser.report will appear. Open that file, seen below:

```
Initial value of k is 13.
When k is 13 0.878668690267916 of the kmers from the median length sequence are unique.
When k is 15 0.978245900878365 of the kmers from the median length sequence are unique.
When k is 17 0.987923309314336 of the kmers from the median length sequence are unique.
When k is 19 0.988980568193194 of the kmers from the median length sequence are unique.
When k is 21 0.989249938982288 of the kmers from the median length sequence are unique.
When k is 23 0.989423053201633 of the kmers from the median length sequence are unique.
When k is 25 0.989563160531508 of the kmers from the median length sequence are unique.
When k is 27 0.989692481570309 of the kmers from the median length sequence are unique.
When k is 29 0.989815852535779 of the kmers from the median length sequence are unique.
The optimum value of K is 31.
When k is 31 0.990570827472672 of the kmers from the median length sequence are unique.

There were 5 genomes.
The median length genome was 4033464 bases.
The time used was 321 seconds

From a sample of 990 unique kmers  304 are core kmers.
0.307070707070707 of the kmers are present in all genomes.

FCK = 0.307.
```

Kchooser assumes that no more that 1% of the genome is duplicated sequences, and keeps incrementing the kmer size until it estimates that 99% of the median length genome is unique. In this case it appears that a plateau value of a bit more than 0.985 is reached when k is 17, so a setting of 98.5% unique sequence would be better. Thus, we will re-run Kchooser and include the optional argument for fraction of unique sequences.

Enter `Kchooser VC.fasta 0.985.` The new Kchooser report shows that the optimum kmer size is 19.

```
Initial value of k is 13.
When k is 13 0.878668690267916 of the kmers from the median length sequence are unique.
When k is 15 0.978245900878365 of the kmers from the median length sequence are unique.
When k is 17 0.987923309314336 of the kmers from the median length sequence are unique.
The optimum value of K is 19.
When k is 19 0.989459988977695 of the kmers from the median length sequence are unique.

There were 5 genomes.
The median length genome was 4033464 bases.
The time used was 179 seconds

From a sample of 991 unique kmers  342 are core kmers.
0.34510595358224 of the kmers are present in all genomes.

FCK = 0.345.
```

Note the bottom line of the Kchooser report. It shows that the fraction of kmers that are present in all genomes (FCK) is 0.334. FCK is a measure of sequence diversity, the lower is FCK the more diverse are the

sequences. As diversity increases the efficiency with which kSNP3 detects SNPs decreases (Gardner & Hall, 2013). Simulation studies (Hall, 2015 Cladistics **32:** 90-99 have shown when FCK is ≥ 0.1 SNP detection efficiency is adequate, and the accuracy of parsimony trees estimated by kSNP3 is > 97%; i.e. the trees can be considered to be reliable.

Based on Kchooser, we suggest a kmer size of 19.

At this point you are done with the VC.fasta file, so delete it to save drive space

Finally, after all that, run kSNP3 by entering:

```
kSNP3 -in in_list -outdir Run1 -k 19 -annotate annotated_genomes | tee Run1Log
```

On my computer Run1 required 20.4 minutes.

Now we will mimic a situation in which we need to add another sequence to the kSNP3 analysis. Perhaps the new sequence just became available, perhaps we would like to add an outgroup for purposes of rooting the tree. The new sequence is listed in the file in_list2, which you suitably modified to include the correct path.

The command line will be just the same as above except that we will use in_list2 as the input file, and we will add the *path* to the existing SNPs_all file from the previous run. Where is the SNPs_all file? Sitting happily in the Run1 directory along with all of the other output files that kSNP3 wrote. We need to add the option -SNP_all to the previous command line. Enter

```
kSNP3 -in in_list2 -outdir Run2 -k 19 -annotate annotated_genomes -SNPs_all #path to the SNPs_all file in the Run1 directory#| tee Run2Log
```

Note: if any of the directories (folders) in that path have spaces in the directory name be sure to enclose that directory name in quotes (" ") in the path.

On my computer Run2 required only 2.9 minutes, considerably better than it would have taken to start over with all six sequences.

V The output files

Most of the output 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.

.

VI. The kSNP3 Utilities

In addition to the executables (programs) that make up kSNP3 itself there are several "utility" programs that are designed to make your life easier. You have already encountered four utility programs: ***MakeKSNP3infile, genome_names3, Kchooser,*** and ***MakeFasta.*** This section discusses those programs, plus several others, 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.

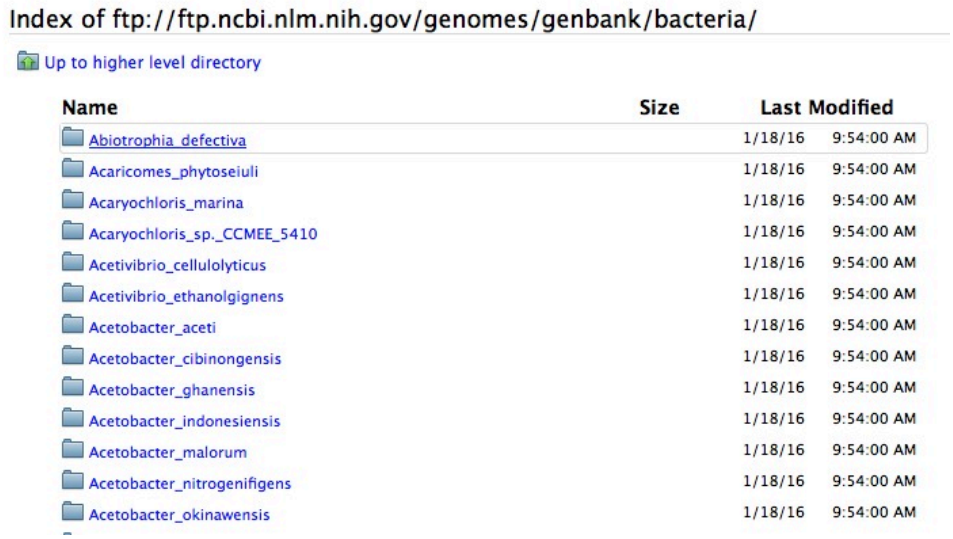
Utilities to download genomes

One of the most tedious parts of comparing genomes is downloading the genome sequence files from the databases. Two utilities, ***parse_assembly_summary*** and ***FTPgenomes*** automate that process.

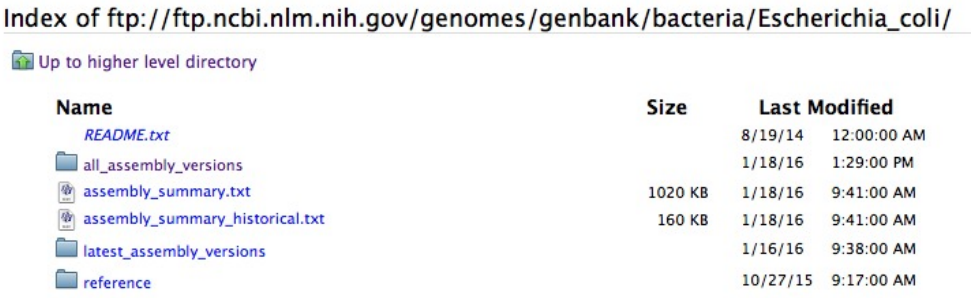
parse_assembly_summary

Genome sequences can be downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/. Within that site there are a series of directories, one for each species (sometime only one isolate is present)

In your favorite browser (other than Safari!!!) go to that site which will look like this figure below.



Scroll down to the directory (folder) for the species of interest. I'll use Escherichia coli as an example. Double-click the Escherichia_coli directory to open it to see:



Double-click the file named "assembly_summary.txt" to open it. Ignore any warnings about not being able to find the file.

That file lists every genome in the directory and gives 19 columns of information about each genome. Don't worry, you don't have to read it.

Save the file as a text file to a suitable folder on your drive by choosing Save Page As… from the file menu and choosing a suitable folder to save the file. The Save dialog will automatically assign the name "assembly_summary.txt", but I suggest modifying the name to something like "Ecoli_assembly_summary.txt" to distinguish it from similar files for other species. That file is just a text file that can be read with any text editor such as BBEdit or TextWrangler (Mac) or whatever is your favorite Linux text editor. The file has a plethora of information about each genome file, but you don't really need to see that information unless you are just curious.

The program ***parse_assemble_summary*** simplifies that file down to something more manageable. In the Terminal program navigate to the folder where you saved the assemble summary file then enter (using the E. coli example):

```
parse_assembly_summary Ecoli_assembly_summary.txt EcoliGenomesList.txt
```

parse_assemble_summary will simplify the file and save the simplified file under whatever name you chose, in this case EcoliGenomesList.txt as shown below

organism_name	assembly_level	ftp_path
Escherichia coli str. K-12 substr. MG1655	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000005845.2_ASM584v2
Escherichia coli O157:H7 str. EDL933	Chromosome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000006665.1_ASM666v1
Escherichia coli CFT073	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000007445.1_ASM744v1
Escherichia coli O157:H7 str. Sakai	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000008865.1_ASM886v1
Escherichia coli BL21(DE3)	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000009565.2_ASM956v1
Escherichia coli str. K-12 substr. W3110	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000010245.1_ASM1024v1
Escherichia coli SE11	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000010385.1_ASM1038v1
Escherichia coli SE15	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000010485.1_ASM1048v1
Escherichia coli O103:H2 str. 12009	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000010745.1_ASM1074v1
Escherichia coli O111:H- str. 11128	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000010765.1_ASM1076v1
Escherichia coli UTI89	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000013265.1_ASM1326v1
Escherichia coli 536	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000013305.1_ASM1330v1

The parsed file has only 3 columns of information about each genome. The first column is the genome name, the second gives some information about state of the genomes file (Chromosome, Complete Genome, Scaffold, or Contig), the third column is the path to the directory on the ftp site where the file you need is located.

You only need to download the assembly-summary file and parse it once for each species you are interested in (unless you would like to update the files). The parsed file becomes the list of genomes from which you will copy lines for each genome you want to download to make the input file for the ***FTPgenomes*** program. You can (and probably should) discard the assembly_summary file

FTPgenomes

Suppose that you would like to download all of the genomes of serotype O157:H7 (the famous Jack-in-the-Box outbreak serotype).

Make a new empty text file called something like O157H7.in to use as the input file for ***FTPgenomes.***
Do a global search in the parsed file for O157:H7 in the simplified EcoliGenomesList.txt file, and copy every line with O157:H7 in the genome name and paste that line into the file O157H7.in. (There are actually 117 such genomes, so below I'll only show a few of them).

```
Escherichia coli O157:H7 str. EDL933    Chromosome          ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000006665.1_ASM666v1
Escherichia coli O157:H7 str. Sakai     Complete Genome     ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000008865.1_ASM886v1
Escherichia coli O157:H7 str. EC4115    Complete Genome     ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000021125.1_ASM2112v1
Escherichia coli O157:H7 str. TW14359   Complete Genome     ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000022225.1_ASM2222v1
Escherichia coli O157:H7 str. EC4024    Scaffold            ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000155005.1_ASM15500v1
Escherichia coli O157:H7 str. TW14588   Chromosome          ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000155125.1_ASM15512v1
Escherichia coli O157:H7 str. EC4196    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171915.1_ASM17191v1
Escherichia coli O157:H7 str. EC4113    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171935.1_ASM17193v1
Escherichia coli O157:H7 str. EC4076    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171955.1_ASM17195v1
Escherichia coli O157:H7 str. EC4401    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171975.1_ASM17197v1
Escherichia coli O157:H7 str. EC4486    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171995.1_ASM17199v1
Escherichia coli O157:H7 str. EC4501    Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172015.1_ASM17201v1
Escherichia coli O157:H7 str. EC869     Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172035.1_ASM17203v1
Escherichia coli O157:H7 str. EC508     Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172055.1_ASM17205v1
Escherichia coli O157:H7 str. FRIK966   Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000175735.1_ASM17573v1
Escherichia coli O157:H7 str. FRIK2000  Contig              ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000175755.1_ASM17575v1
```

The 3rd (right-most) column is the path to the genome sequence file at the FTP site. The first (left) column is the name that will be assigned to the file when it is downloaded to your drive. Those genome names are pretty long for file names and they may contain some characters that your operating system will not allow in file names, so it is worthwhile to edit the genome names if you like. Use the global search & replace function of your text editor to do that. I sequentially replaced "Escherichia coli ", and ":" with nothing, and replaced "str. " with '_' to produce the edited file shown below

```
O157H7_EDL933           Chromosome          ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000006665.1_ASM666v1
O157H7_Sakai            Complete Genome     ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000008865.1_ASM886v1
O157H7_EC4115           Complete Genome     ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000021125.1_ASM2112v1
O157H7_TW14359          Complete Genome     ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000022225.1_ASM2222v1
O157H7_EC4024           Scaffold            ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000155005.1_ASM15500v1
O157H7_TW14588          Chromosome          ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000155125.1_ASM15512v1
O157H7_EC4196           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171915.1_ASM17191v1
O157H7_EC4113           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171935.1_ASM17193v1
O157H7_EC4076           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171955.1_ASM17195v1
O157H7_EC4401           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171975.1_ASM17197v1
O157H7_EC4486           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000171995.1_ASM17199v1
O157H7_EC4501           Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172015.1_ASM17201v1
O157H7_EC869            Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172035.1_ASM17203v1
O157H7_EC508            Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000172055.1_ASM17205v1
O157H7_FRIK966          Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000175735.1_ASM17573v1
O157H7_FRIK2000         Contig              ftp//ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000175755.1_ASM17575v1
```

In Terminal navigate to the folder containing the input file and enter `FTPgenomes infileName`, or in this example `FTPgenomes O157H7.in`.

```
Scorpion Example barry$ FTPgenomes O157H7.in
Downloading GCA_000006665.1_ASM666v1_genomic.fna.gz ...
Downloading GCA_000008865.1_ASM886v1_genomic.fna.gz ...
Downloading GCA_000021125.1_ASM2112v1_genomic.fna.gz ...
Downloading GCA_000022225.1_ASM2222v1_genomic.fna.gz ...
Downloading GCA_000155005.1_ASM15500v1_genomic.fna.gz ...
Downloading GCA_000155125.1_ASM15512v1_genomic.fna.gz ...
Downloading GCA_000171915.1_ASM17191v1_genomic.fna.gz ...
Downloading GCA_000171935.1_ASM17193v1_genomic.fna.gz ...
Downloading GCA_000171955.1_ASM17195v1_genomic.fna.gz ...
Downloading GCA_000171975.1_ASM17197v1_genomic.fna.gz ...
Downloading GCA_000171995.1_ASM17199v1_genomic.fna.gz ...
Downloading GCA_000172015.1_ASM17201v1_genomic.fna.gz ...
Downloading GCA_000172035.1_ASM17203v1_genomic.fna.gz ...
Downloading GCA_000172055.1_ASM17205v1_genomic.fna.gz ...
Downloading GCA_000175735.1_ASM17573v1_genomic.fna.gz ...
Downloading GCA_000175755.1_ASM17575v1_genomic.fna.gz ...
Used 145.369590044 seconds to download these files.

Scorpion Example barry$
```

The files will be downloaded into the folder that contains the input file. You can move them to wherever you want to keep them permanently and discard the input file.

Utilities to use before running kSNP3

FG2IF

Converts a fasta file of many genomes to individual fasta genome files, collecting them in an output directory that you specify.

This utility is designed for kSNP v2 users who may already have a data set in the form of a single fasta file.
Usage: `FG2IF myGenomes.fasta outputDirectory`

After running FG2IF the resulting directory should be moved to its final location.

genome_names3

Extracts the genome names from a kSNP3 input list such as in_list and writes a file that lists the genome names.
Usage: `genome_names3 in_list genomeNames`

To generate a list of annotated genomes first add all of the annotated genomes to in_list, then run genome_names3 using an output file name such as annotated_genomes, then add all of the unannotated genomes to in_list.

Kchooser

Kchooser 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 kSNP3 is likely to identify SNPs from your data set and the accuracy of parsimony trees at that optimum k.

To estimate the optimum k, Kchooser determines the fraction of kmers that occur only once in the median-length genome from your data set. Some kmers will occur more than once because of genuine duplicated regions, so Kchooser looks for a value of k such that at least 0.99 of kmers are unique. The expectation is that less than 1% of a real genome will be duplicated.

Kchooser 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 the fraction of core kmers decreases and the efficiency with which kSNP3 detects SNPs decreases. Simulation studies (Gardner & Hall, 2013; Hall 2015 Cladistics **32:** 90-99) have shown that efficiency is >97%, and the accuracy of parsimony trees is > 0.97 when the fraction of core kmers reported by Kchooser is ≥ 0.1.

The input to Kchooser is the fasta file of genome sequences for your data set. Use the utility program MakeFasta to make the fasta file from a *copy* of the kSNP3 input_list file. **Bes sure to remove from that copy any genomes based on raw reads!!** See Section VI, kSNP3 utilities under Utilities to use before running kSNP.

Usage: Kchooser myfile.fasta [fraction unique kmers]
Example: `Kchooser fastainput`.
Output file: Kchooser.report (**Kchooser** saves the output file named Kchooser.report. You can rename it later if you wish).

Optional: in some cases the genomes actually do have more than 1% duplication. In that event Kchooser will continue increasing k until it reaches a maximum of k=31. In that event the Kchooser.report file may look something like this

```
When k is 13 0.81352773629423 of the kmers from the median length sequence are unique.
When k is 15 0.943884435193731 of the kmers from the median length sequence are unique.
When k is 17 0.958739180241411 of the kmers from the median length sequence are unique.
When k is 19 0.960580220065033 of the kmers from the median length sequence are unique.
When k is 21 0.961173052710233 of the kmers from the median length sequence are unique.
When k is 23 0.961571726800186 of the kmers from the median length sequence are unique.
When k is 25 0.961914216402944 of the kmers from the median length sequence are unique.
When k is 27 0.962221869519791 of the kmers from the median length sequence are unique.
When k is 29 0.962500075446101 of the kmers from the median length sequence are unique.
The optimum value of K is 31.
```

The fraction of unique kmers has reached a plateau at a bit over 0.96, suggesting that almost 4% of that particular genome is duplicated. Kchooser provides the option of setting a lower value for the unique kmer cutoff. In this case an appropriate cutoff would be 0.96.
Example: `Kchooser fastainput 0.96`

Kchooser runs fairly quickly. For a data set consisting of 68 *E. coli* genomes Kchooser required less that 10 minutes. For a data set of 119 *E. coli* genomes that included two genomes as raw reads Kchooser used 62 minutes. The increase is the result of the time it takes to check the very long raw sequences. To speed up the process raw read genomes can be removed from the fasta file before running Kchooser.

MakeFasta

Although kSNP3 does not use a fasta file for its input, Kchooser requires a fasta file in order to estimate the optimum kmer size (-k argument in kSNP3) and to estimate FCK (the fraction of core kmers) which is important for predicting the reliability of phylogenetic trees estimated by kSNP3. ***MakeFasta*** uses the same input file (list of individual fasta genome files and the paths to those files) as does kSNP3 to write a single fasta file of all of the genome sequences except those based on raw reads. That file can be used as the input to Kchooser. After running Kchooser, unless you foresee a further use for it, that fasta file should be deleted to save hard drive space.

Usage: `MakeFasta infile_name outfile_name`
infile_name is the name of the file that lists the genome sequence files; i.e. the kSNP3 input file
outfile_name is the name of the output fasta file
Example: `MakeFasta in_list Eco68.fasta`

MakeKSNP3infile

Makes a kSNP3 input file (the input list of genomes) from a set of all of the genome fasta files within a directory. The directory must contain no other files.

Semi-automatic mode: the input list that is written includes the path to each fasta genome file, but the user must manually enter a name for that genome. The mode option = S.

Automatic mode: the input list that is written includes both the path to each genome file and the name for the genome, when the name is derived from the file name. *The file name must not contain any spaces*. The mode option = A.

Navigate to the directory that *encloses* the directory of interest.

Usage: `MakeKSNP3infile directory outfileName mode`
Examples: `MakeKSNP3infile Ecoli Eco115 A` (automatic mode)
`MakeKSNP3infile Ecoli Eco115 S` (semi-automatic mode)

merge_fasta_reads3

Assume that you have a file of raw reads with the name g2_reads.fasta
Enter: `merge_fasta_reads3 g2_reads.fasta > g2_merged.fasta`
g2_merged.fasta is the name of the file of the merged genome.

Utilities to use after running kSNP3

extract_nth_locus

This utility extracts locus number n from a SNPs file.

The command is:
`extract_nth_locus n (SNPs_all` or `core_SNPs` or `SNPs_in_majority0.5) > outfileName`

Examples:

```
extract_nth_locus 5 core_SNPs > locus_5_in_core_SNPs
extract_nth_locus 155 SNPs_all > locus_155_in_SNPs_all
extract_nth_locus 30 SNPs_in_majority0.5  > locus_30_in_SNPs_in_majority
```

rm_node_names_from_tree3

rm_node_names_from_tree removes the labels from internal nodes
Usage:

`rm_node_names_from_tree 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

NodeChiSquare2tree31

NodeChiSquare2tree31 identifies for each node the SNPs for which the χ^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)

NodeChiSquare2tree31 is run from with the kSNP3 output file directory after kSNP3 has run.

Usage: `NodeChiSquare2tree31 [-p maximumChiSqProbability -t treeType]`
-p maximum Chi Sq probability for assigning a SNP to a node.
-t tree type from kSNP3 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.

select_node_annotations3

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 `select_node_annotations3 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 `select_node_annotations3` without any arguments instructions will be displayed on the screen.

Utilities to troubleshoot kSNP3

check_genbank_from_NCBI

Problem: during annotation kSNP3 fails with an error message that includes "ParAnn returned -1".

During the annotation process kSNP3 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 kSNP3 will crash during the annotation process with an error message that includes "ParAnn returned -1".

The utility check_genbank_from_NCBI 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 kSNP3 output files. Enter `check_genbank_from_NCBI`. 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 kSNP3 output files, including the modified genbank_from_NCBI file. Enter `ParAnn` to restart the annotation process and complete the kSNP3 analysis.

fix_old_fasta_headers

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 kSNP3 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:
`fix_old_fasta_headers targetFolderName`

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

VII. Installing kSNP3

> 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.

kSNP3 consists of a Unix shell script and a set of Perl executables that use four separate third-party programs to accomplish the kSNP3 goals. kSNP3 is provided as separate packages for Mac OS X and for Linux operating systems. Whichever version you chose 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 kSNP3 that contains some 61 files. The kSNP3 directory contains a tcsh script named kSNP3 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 kSNP3 scripts.

The Mac OS X has the tcsh shell installed by default, but some Linux installation do not include tcsh. To see what shell is currently running, enter `echo $0` (that is a zero). It will respond with something like `bash` or `tcsh.` To see if tcsh is installed, enter `tcsh.` If it is installed the prompt will change to % or >. If tcsh is not installed you will get a message informing you of the problem.

Assuming that you are connected to the internet, enter
`sudo apt -get install tcsh`. After a few moments tcsh will download and install. To check that it is installed enter `tcsh` then enter `echo $0`. The response should be `tcsh`. To get back to bash (if that was what was running) just enter `bash`. kSNP3 works just fine from the bash shell, it just requires that tcsh is installed - not currently running.

kSNP3 should be installed into the /usr/local directory, and once installed the kSNP3 directory should not be moved afterwards. It is possible to install kSNP3 elsewhere, but if you do so you will need to modify the kSNP3 file itself to point to the directory where you have installed kSNP3 (see the top of the kSNP3 file). If you do not know how to do that, then either install kSNP3 into /usr/local as explained below, or get some to help you correctly modify the kSNP3 file after installing it elsewhere.

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 kSNP3 directory from the kSNP3.0_Mac_package directory into the usr/local directory. You may be notified that kSNP3 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: navigate to the kSNP3.0_Linux_package directory that contains the kSNP3 directory then enter `sudo mv kSNP3 /usr/local`. You may be asked for a password, if so enter it.

To install kSNP3 into an unprotected directory just drag the kSNP3.0 directory to the chosen directory, but note the instructions above concerning installation of kSNP3.0 into a directory other than /usr/local.

VIIA Set the PATH variable

The operating system needs to know exactly where all the programs in kSNP3 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; under the tcsh shell it is called .tcshrc. In both cases the file is invisible because its name begins with a dot. In **Linux**, in your home directory under the **View** menu choose **Show hidden files**. In **Mac OS X**, in Terminal enter `defaults write com.apple.Finder AppleShowAllFiles YES`. You should now see the hidden files. If you do not see a .bash_profile or a .tcshrc 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 kSNP3 into usr/local. If you installed kSNP3 elsewhere add the path to the installed location instead.

bash shell: If the .bash_profile file does not exist, make a new text file named .bash_profile (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/kSNP3:$PATH"`

If the .bash_profile file already exists and includes an `export PATH` line, perhaps something like `export PATH="/usr/local/ActivePerl-5.10/bin:$PATH"` just add `/usr/local/kSNP3:` between the : and the $.

tcsh shell: If the .tcshrc file does not exist, make a new text file named .tcshrc (don't forget the leading dot!), then enter the following line
`setenv PATH $PATH\:/usr/local/kSNP3`

If the .tcshrc file already exists and includes an `PATH` line, perhaps something like

`setenv PATH $PATH\:/usr/local/ActivePerl-5.10/bin,` just add `:/usr/local/kSNP3` to the end of the line.

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

VIII How kSNP3 works

The kSNP3 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).

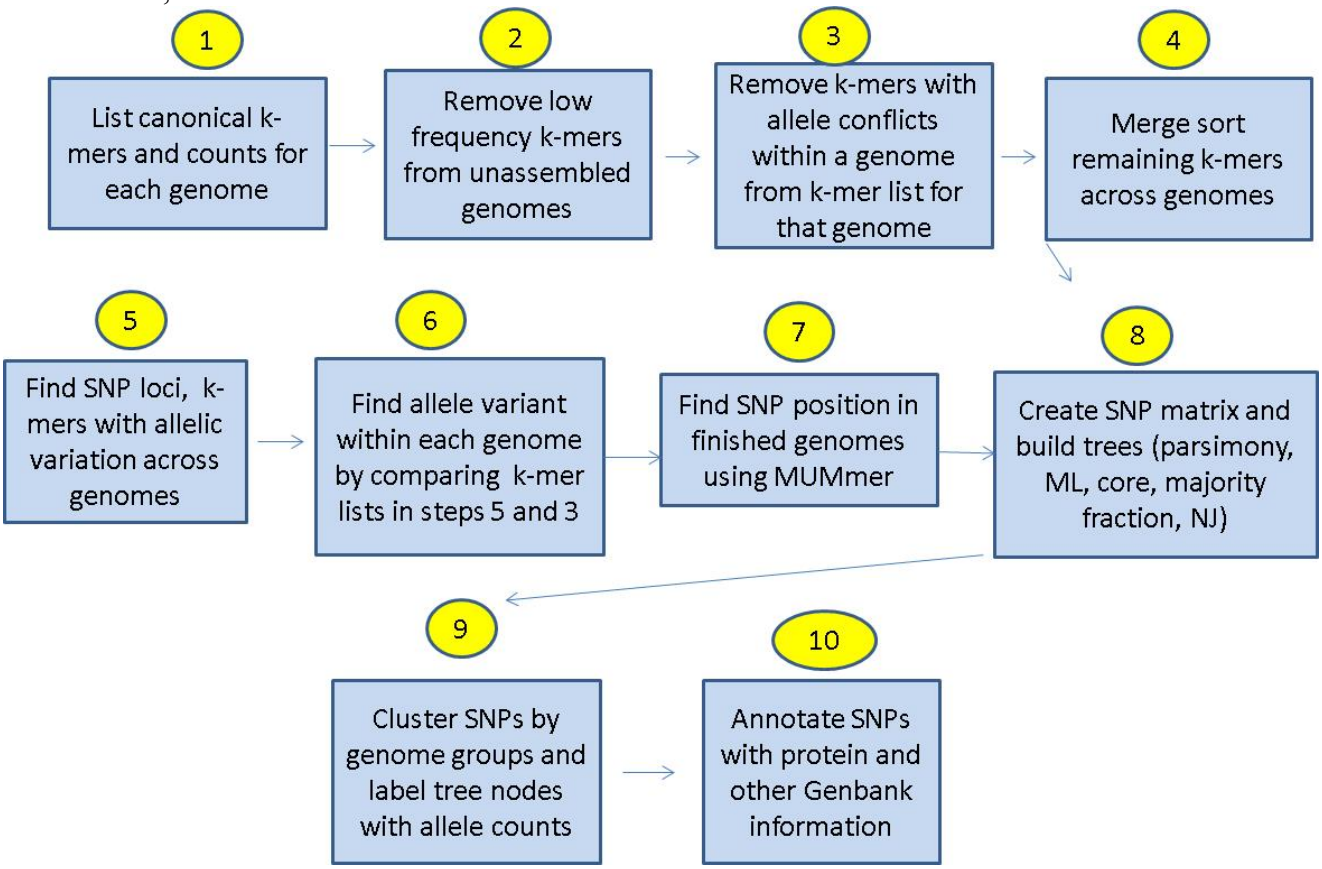


Figure 1: diagram outlining kSNP3 process

IX Time and Efficiency Considerations

The time required to run kSNP3 depends both on the data set and upon the options chosen; i.e. the tasks that kSNP3 is required to carry out. Below we report the time required for kSNP3 to analyze a set of 20 finished *Escherichia coli* genomes and we compare that time with the time required by the earlier version kSNP v2.

Under the default options kSNP3 identifies SNPs and estimates a consensus parsimony tree. All runs were done on a Macintosh iMac with a 3.7 GHz Intel core i7 processor and 16 GB RAM.

Program	**Conditions**	**Time (hours)**
kSNP3	default (no annotation)	0.89
kSNP v2	default (no annotation)	1.04
kSNP3	annotation	2.92
kSNP v2	annotation	11.04
kSNP3	default plus ML **or** NJ **or** VCF	0.97 **or** 0.93 **or** 0.89

Annotation increases run time significantly. If you are not interested in identifying SNP positions in each genome or the genes within which the SNPs lie there is no reason to include the -annotate option.

X Citations

If you use kSNP3 or kSNP v2, 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.

kSNP3 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 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.

XI Licenses

kSNP3
by Shea N. Gardner

kSNP3 OCEC-12-091

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.

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.

XII Frequently Asked Questions

Who should I contact for support of kSNP3?

Unfortunately there is no "support team" for kSNP3 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 kSNP3? If not, please download and install the most recent version which is currently v3.0, 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 kSNP3 from the source code we can offer no support whatsoever. Download one of the executable packages and try that.
3. Did you install the kSNP3 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 kSNP3?

The utilities **FetchFinishedGenomes** and **FetchGenomeAssemblies** are designed to simplify and automate the process of download large sets of genomes. See VI. The kSNP3 Utilities section under Utilities to download genomes for details and use of those programs.

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 v2 from my computer in order to run kSNP3?

No, you can keep both versions if you want to. The earlier version will not interfere with kSNP3.

The only reason we foresee to keep kSNP v2 is to be able to use a fasta file of multiple genomes that you already have on hand. Because kSNP3 is faster and because it properly annotates SNPs in multiple replicons in the same genome, we think that kSNP3 is a better tool. To facilitate migrating your existing data set for use with kSNP3 we have included the utility ***FG2IF*** (see Section VI The kSNP3 Utilities under Utilities to use before running kSNP3.

How do I uninstall kSNP v2 from my computer?

Simply delete the kSNP directory from /usr/local. After that remove kSNP from your path variable. (see Section VIIA).

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

We have provided a little utility as a tcsh shell script for exactly this purpose. The utility is named **select_node_annotations**. See VI. The kSNP3 Utilities section under Utilities to run after running kSNP3.

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). See Section IX Time and Efficiency Considerations for a discussion of this issue.

How much RAM does kSNP3 require?

Again, that is a function of the data set. On a Linux installation with 48 GB of RAM kSNP3 has never required more than the available RAM. On an iMac with 16 GB of RAM a data set with 119 *E. coli* genomes, including 68 finished genomes, kSNP3 has come close to exhausting the available RAM. It is also a function of the operating system. The near-exhaustion of RAM on the iMac occurred using OS

X 10.7. Memory management under OS X 10.9 is much improved and with the same data set there was no sign of memory exhaustion.

How can I extract the nth locus from the core_SNPs file?

The tcsh script extract_nth_locus does exactly what you want. See VI The kSNP3 Utilities section under Utilities to run after running kSNP3.

I already have downloaded directories of genomes, with each directory containing several .fna files that include files for the chromosome and several plasmids. How can I combine those files into a single fasta file that can be used by kSNP3?

Use the `cat` command. Assume that the directory includes three files, NZ098233.fna, NZ098234.fna, and NZ098235.fna and you want to combine them into a file named Saureus_MP2722.fasta. On the command line enter

```
cat 'NZ098233.fna' 'NZ098234.fna' 'NZ098235.fna'  >  'Saureus_MP2722.fasta'
```

Don't forget the single-quote marks around each file name.

After the file Saureus_MP2722.fasta appears you can move it into another directory, perhaps one with other S. aureus files, if you wish.

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

The utility rm_node_names_from_tree does exactly what you want. See VI. The kSNP3 Utilities in the section Utilities to use after running kSNP3

Can I edit the genome IDs in an in_list file written by MakeKSNP3infile 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 kSNP3 with the -annotate option, can I simply use annotate_SNPs_from_genbankFiles3 with the -all 1 option to do annotation after the fact?

No. annotate_SNPs_from_genbankFiles3 requires some files that are written when -annotate is invoked. You need to repeat the run with the -annotate option, using the -all_annotations option if you want full annotation.

Can I use raw-read files in the fastq format for kSNP3?

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.

• jEdit is another excellent free text editor, and the Java-based version will run on any platform. Go to http://www.jedit.org/index.php?page=download and choose the Java-based installer.

• KomodoEdit is a free text editor from Active State and is available for Linux, and Macintosh (http://www.activestate.com/komodo-edit/downloads).

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.