Institute of Infection and Global Health

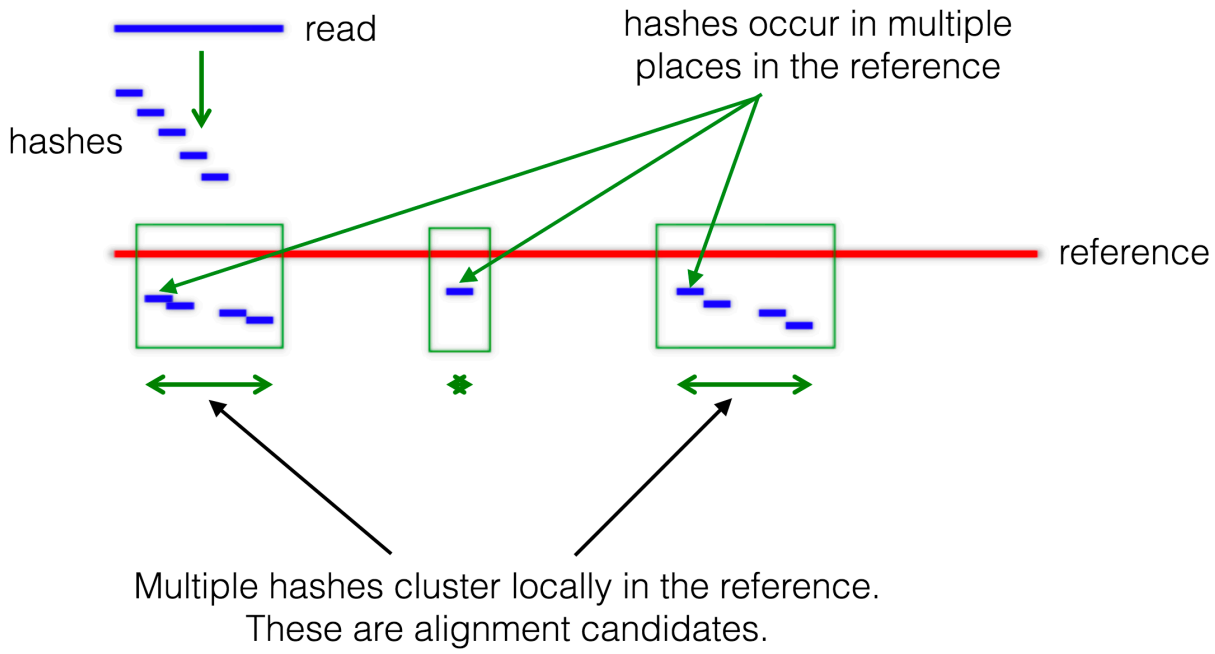
Introduction to sequence informatics

An introduction to Mapping and Population Genetics

During session 2 we introduced you to the basics of next generation sequencing and the process of de novo assembly using Illumina read data. As we discussed, this process requires no prior information about the genome, but often has trouble assembling entire chromosomes.

During this session we will use read mapping, an alternative approach that relies on the use of a reference sequence. Ideally, this reference sequence will have been extensively checked for errors and will include extensive annotation. Widely studied organisms typically have at least one high quality reference sequence, however, for non-model organisms it is common for references to be incomplete or fragmented.

Conceptually, mapping is a similar process to de novo assembly. First, we generate our hash libraries, one for the reference and one for the reads. We then attempt to map the reads to the reference by finding clusters of shared hashes. Compared to de novo assembly this process is much simpler. This is because the number of searches made (millions of reads against tens of reference sequences) is significantly smaller than with a de novo assembly (millions of reads against millions of reads).

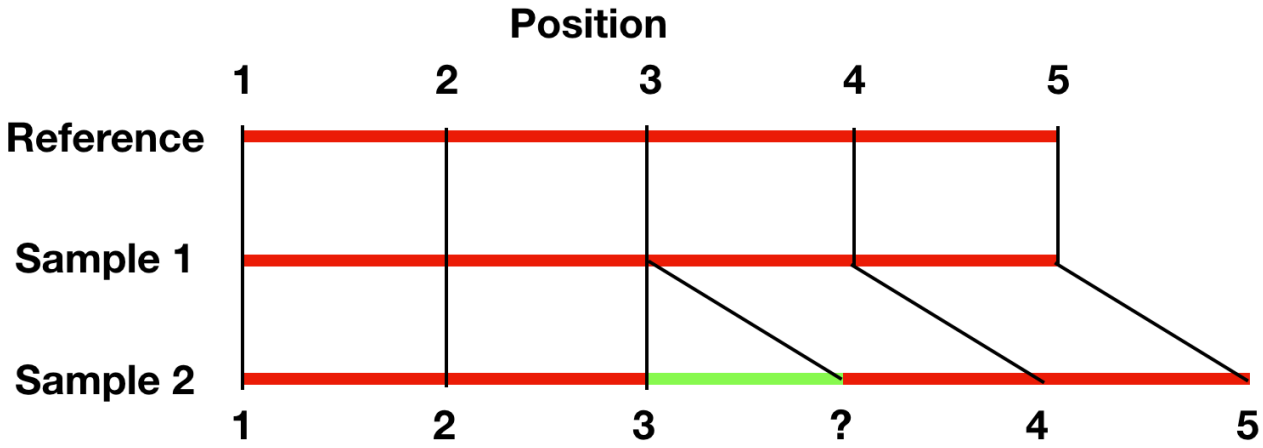


Mapping has both advantages and disadvantages compared to de novo assembly.

Computationally, it is significantly easier. Rather than having to find all the possible reads that might assemble together into a contig, resolve all the possible combinations, and eventually join up individual contigs, we simply have to match the location of each read to the reference sequence.

Mapping also provides the advantage of having each and every sample have a common point of reference. We know that base 159 refers to the same position in sample 1 as it does in sample 3219. By examining the reads which map across this region we can determine if the base is the same or different from that of the reference. This is the basis for SNP calling, which you will perform during the course of this session.

The primary disadvantage of mapping over assembly is the assumption that the reference sequence being used is representative of the sample we are mapping to it. Mapping will either find a location on the reference to position the reads to or discard them entirely. For core genes, this is not often an issue but for hyper-variable, repetitive or telomeric regions the quality of the mapping may often fail or give incorrect results. For example the VSG genes of *Trypanosoma brucei*, which undergo frequent recombination, rarely map to the VSG genes of the *T. brucei* 927 strain reference sequence as they are too divergent from one another.



Insertions, deletions, duplications and rearrangements are also harder to identify using a mapping approach. Novel insertions (such as the green region in sample 2, above) relative to the reference are removed while deletions just appear as sequence with zero coverage. For other insertions, such as a duplication of a gene, the reads from both genes may map on top of one another and effectively hide the duplication. Rearrangements will run into a similar issue, with the software simply forcing the sequence to fit the reference. While it is possible to identify when these events have occurred the process is typically time consuming and introduces its own problems.

Exercise One

For this exercise we will use sequencing data from Ebola. We will map it to two different Ebola references, call SNPs and generate a consensus sequence based on each reference. We will then align these sequences against 36 other sequences and use this alignment to construct a phylogenetic tree to investigate how each of these isolates are related to one another.

1. Log on to the server and copy the session 4 folder to your home directory.

2. Examine the files that are present. What format are they in? Do you recognise the formatting of any of them? The file list should contain two reference sequences (ref1.fa and ref2.fa) in addition to a set of compressed paired-end reads (reads_1.fastq.gz and reads_2.fastq.gz)

3. We will be using a widely used piece of software called bwa to map the reads to both references during this exercise. Throughout the exercise we will provide the commands to map the reads to the first reference file but you will need to alter them to repeat the process for mapping to the second reference.

   The first step is to index the reference file. This will create the hash library and allows bwa to rapidly locate sequence motifs within the reference genome. The command is:

   bwa index ref1.fa

4. Next, we will map the reads to the reference using the bwa mem algorithm. As we are using paired end reads bwa will take this information into account when identifying where to place each read. Locations with partial matches may, therefore, be chosen if it maintains the read pair even if there is a better match for one read elsewhere in the genome. Bwa will output the results in the sam format, which stands for Sequence Assembly/Mapping. The command for this is:

   bwa mem ref1.fa reads_1.fastq.gz reads_2.fastq.gz > ref1.sam

5. Once you have created the sam file open it and examine the contents. Using what you have already learned find the reads M01760:76:000000000-AE8GG:1:2114:8465:23918 and work out their bitwise flag values. What do these two values mean? You will need to search online for how to do this.

6. While sam files are human readable they are a large and inefficient (as far as the computer is concerned) format so we will now convert the sam file to a bam file, which is a compressed binary version of the sam format. Convert the sam to bam using:

   samtools view -T ref1.fa -Sb ref1.sam > ref1.bam

7. The sam file that you have created is quite large. Remove it using the appropriate command from the introduction to linux session.

8. When we created the sam file the reads within it were sorted simply by their individual identifier, read 1 followed by read 2, followed by read 3 etc. It is extremely unlikely that these reads have mapped to the same regions of the genome. We are going to resort the reads based on where they have mapped to in the genome. This will start from base 1 of each chromosome, identify all of the reads mapping to it and then save them to the same part of the bam file. When we open the bam file in future the computer can quickly identify and extract the reads in any given region rather than having to hunt through the entire file. The command for this is:

   samtools sort ref1.bam -o ref1.sorted.bam

9. Remove the unsorted bam file

10. Before we can examine the bam file we need to index it. The command this time is:

    samtools index ref1.sorted.bam

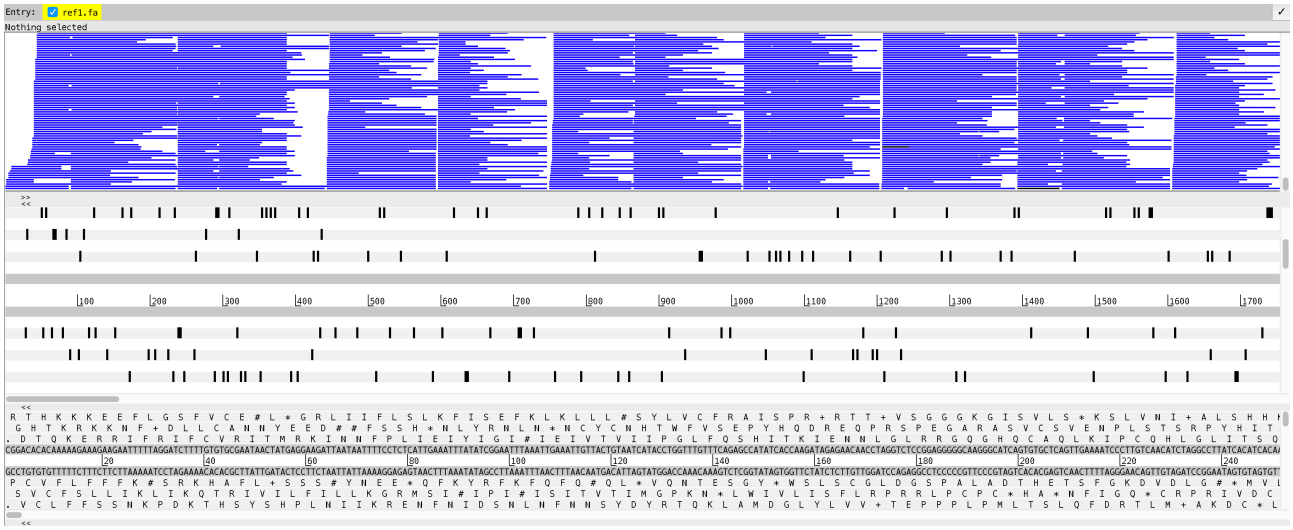
11. Examine the bam files you have created using artemis. The first step is to open artemis and the first reference sequence using:

    art ref1.fa &

12. Load in the bam file that you mapped to ref1 using

    File -> Read BAM / VCF

    You should now have a screen that looks something like this:



    As when we used Artemis to examine the RNA-Seq data during session 2 our top panel represents a pileup of all the reads, aligned based on their mapping to the reference sequence. Each small bar here represents a single read and in the initial view there may be hundreds of reads present.

    Compared to the RNA-Seq data our reads map to the entire genome. Why might this be?

    It is clear that we have a high level of coverage of this genome. What factors may contribute to this? Why might we expect a lower coverage if we were to sequence the human genome using the same amount of effort?

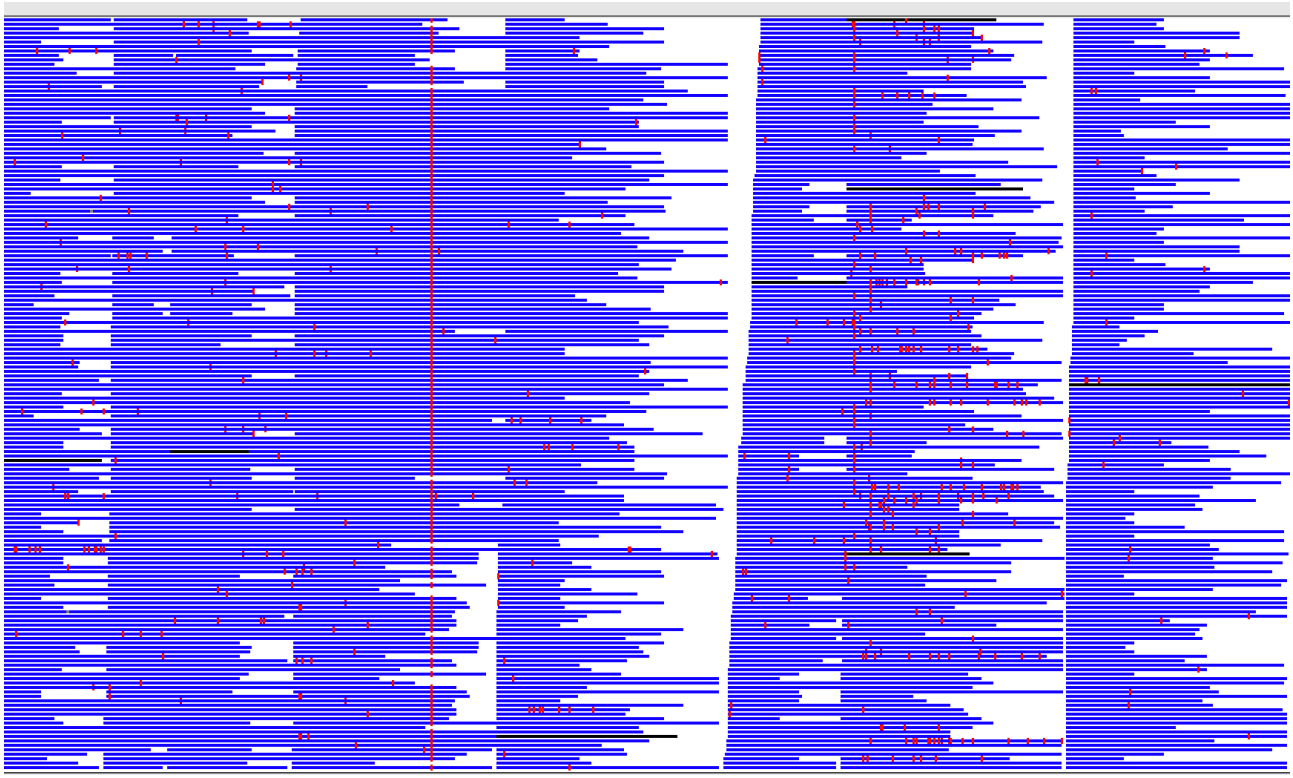
13. Zoom in until Artemis shows you the individual bases of each read. For our population genetic analysis we need to identify the individual bases where our samples differ from the reference. First we are going to try and do this manually. Right click on the reads pane and select:

    Show -> SNP marks

    Individual bases that disagree with the reference will now be highlighted in red. Can you spot any patterns?

Why do you think you are unlikely to find clusters where consecutive bases are marked in red?

14. Zoom out slightly and examine the distribution of the red marks throughout the genome. You should see that the majority of red marks appear to be distributed at random while the remainder either cluster together or form a solid vertical red line.



It's important to remember that our x-axis represents the position on the reference genome. The vertical red line, therefore, indicates that all of the reads mapping to this position disagree with the reference.

In a diploid organism, such as a human, reads originate from one of the two chromosomal homologues. In a pileup view such as this we would, therefore, expect each position to be all blue (homozygote, agrees with the reference), all red (homozygote, disagrees with the reference) or approximately 50% and 50% blue (heterozygote, one homologue agrees with the reference, one disagrees).

Ebola is a single stranded, haploid RNA virus so we might expect all positions to be either 100% blue or 100% red. Why then, do we see putative SNPs that are a mix of red/blue?

15. Close artemis and restart it using reference 2. Load in the bam file, are there any clear differences in the SNP patterns?

16. Now that we have examined the possible SNPs manually we need to identify them computationally. This process will examine the pileup at each position, filter out reads which have mapped with a low quality and then determine if the reads agree or disagree with the reference overall. For example, positions where one read differs will not be called as a SNP if there are 50 reads that agree with the reference.

    First we need to index the reference sequence once again!

    samtools faidx ref1.fa

17. After indexing the reference we are going to use it and the bam file to generate a list of all SNPs in two VCF (Variant Call Format) files. The first, .vcf, is an uncompressed text file that you should examine using less. The second, .vcf.gz, is a compressed version of that file. To generate these files we will be combing two commands using the | (pipe) command, which takes the output from one command and uses it as the input for the second command. There are many different approaches to call SNPs, the one we are using here is relatively basic and not recommended for a real world analysis.

    bcftools mpileup -f ref1.fa ref1.sorted.bam | bcftools call -vc --ploidy 1 -O v > ref1.haploid.vcf

    bcftools mpileup -f ref1.fa ref1.sorted.bam | bcftools call -vc --ploidy 1 -O z > ref1.haploid.vcf.gz

    tabix ref1.calls.vcf.gz

    You may have noticed that we set the --ploidy option to 1. This indicates that we are only interested in calling SNPs where there is a single dominant variant present as the Ebola genome is haploid. It is possible that a single individual could be infected with more than one strain, or that the strain has mutated during the course of the infection.

    Repeat this process for the ref2 files.

    How many SNPs were called in ref1 and ref2? Why might these numbers differ?

18. Return to Artemis and load in your SNP file using

    File -> Read BAM / VCF

    and select the vcf.gz file that corresponds to the current sample. You will see a new pane open that indicates the position of the SNPs you have just called. It should be clear that there are far fewer SNPs than there are red marks. This is because our SNP calling took multiple factors into account when determining if a disagreement with the reference was real or not. These include the base quality of the read, the mapping quality, the uniqueness of the region and even how many other mismatches there are in the local region.

19. To generate the consensus sequence use the following command, which will output a new file in the fasta format.

    bcftools consensus -f ref1.fa -s ref1.sorted.bam ref1.calls.vcf.gz > ref1.cns.fa

    What are the strengths and weaknesses of this approach as opposed to focusing only on the SNPs?

    Our mapping has generated a high level of coverage across the entire genome. This is not always the case. How might we handle the data if we had regions with zero coverage?

    Before you move on to the next exercise ensure that you have generated the consensus sequences for both ref1 and ref2.

Exercise 2

In order to investigate the relationships between the multiple isolates, we need to first align them against one another. You have already run two types of alignment. The first was when you ran blast during the previous sessions, the second time was earlier in this exercise when you mapped your reads to the reference. During the mapping we aligned millions of reads against a single reference sequence. This time we will be aligning a group of 38 Ebola sequences against one another. You have created two of these sequences while the remainder can be found in the file *ebola.aligned*. For the sake of speed we have already aligned these 36 samples against one another

We briefly touched on the difficulty of aligning sequences when insertions and deletions are present. During the mapping we used a fixed reference and effectively ignored indels. During multiple sequence alignment we take the opposite approach, we add gaps in order to account for bases present in one sequence but absent in another. The software will attempt to create the best alignment possible, however, it is not unusual for it to struggle with complex allelic differences, rearrangements or repetitive sequences.

1. The first step for aligning the sequences is to concatenate all of the fasta files into a single multi-sequence fasta file.

   Concatenate your two consensus into a new file called myrefs.fa

2. We are now going to align them against one another using the Clustal Omega (you may have previously used ClustalW, an older version of the software)

   To generate a Clustal Omega alignment run the following command:

   clustalo --threads 1 -i myrefs.fa --p1=ebola.aligned -o allsamples.aligned -t DNA --outfmt=clustal -v

3. The output is a text file containing the alignment. Examine it using less and take particular note of the start, why does it contain so many dashes? We have used the basic alignment parameters for this exercise, in a real world situation you would need to optimise these to obtain the best alignment and even then you may find that you need to manually realign portions of the sequence based on your own knowledge of the sequence.

4. The remainder of the exercise will be completed using R to explore the data. Launch R from the terminal using.

5. For this exercise we are going to use a number of R packages to load additional functions into R. Three of these packages (ape, ips, ggtree) were written specifically for phylogenetic analysis while the remaining two (ggplot2 and gridExtra) extend the plotting functions of R. There are thousands of R packages available, the two repositories you are most likely to require are called CRAN and BioConductor. Once a package has been installed you can load it using the library() function:

   library(ape)
   library(ips)
   library(ggplot2)
   library(ggtree)
   library(gridExtra)

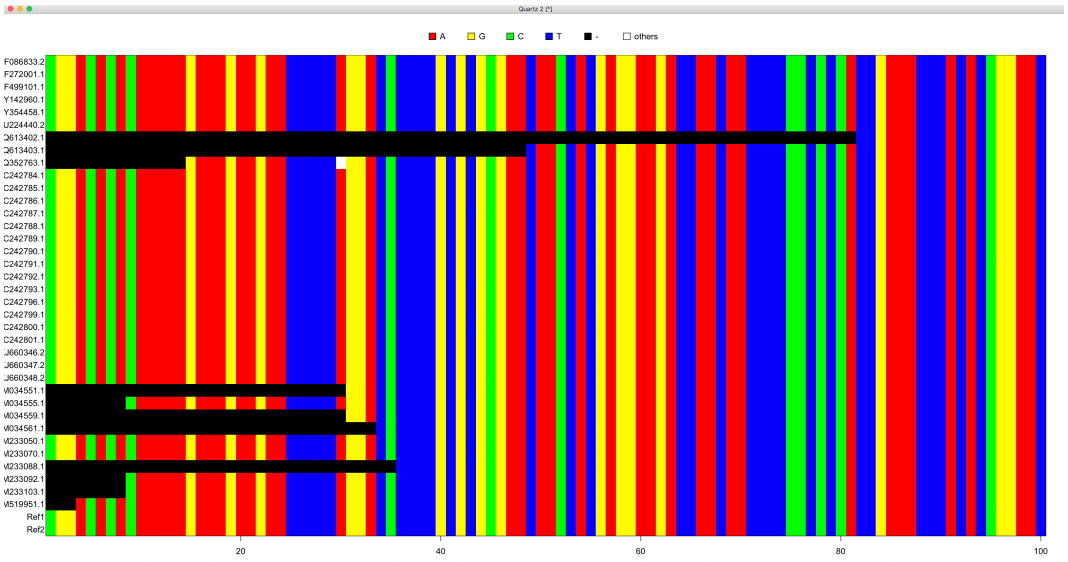
6. The next step is to load in our data

   alignment=read.dna("allsamples.aligned", format="clustal")

   This will store our aligned sequences in the DNAbin format. This is a custom class designed especially for use with the ape package, additional functions within the ape package are able to access the data in a more efficient way than would be possible were it stored in a regular matrix.

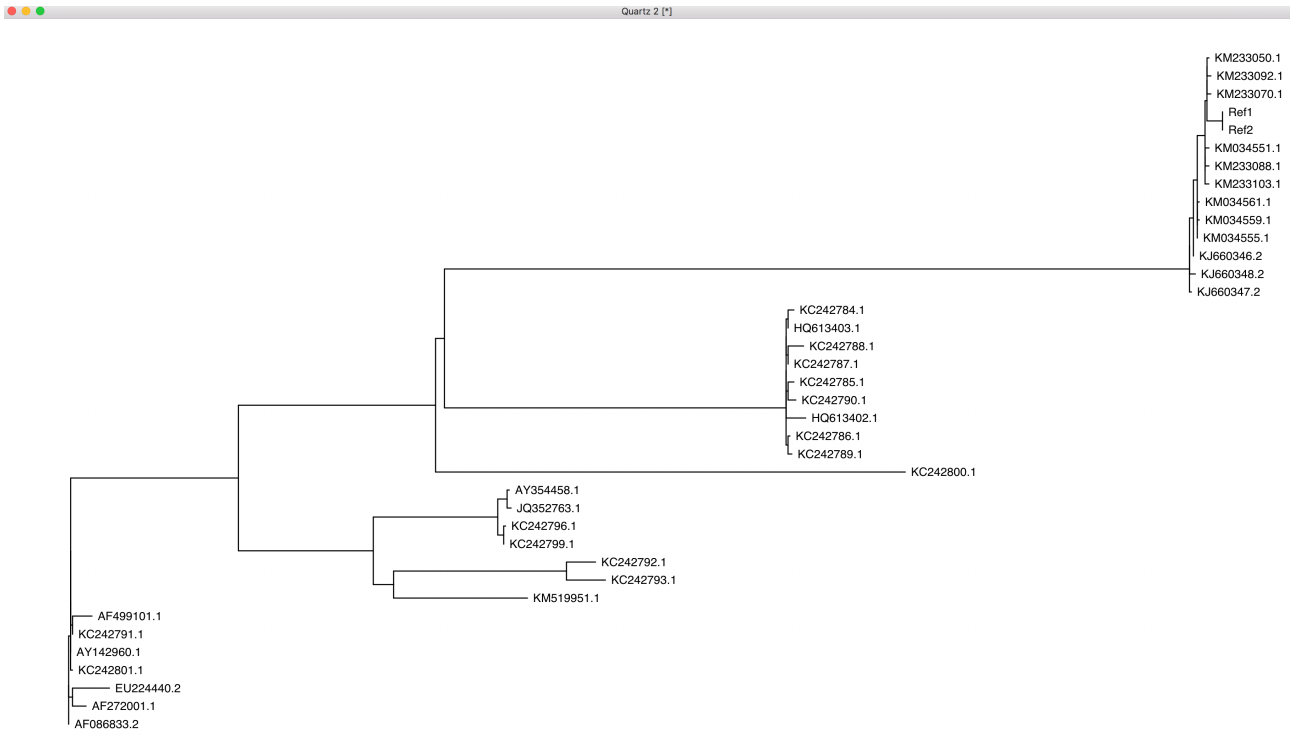
7. For example we could visualise the first 100 bases of the alignment using the image.DNAbin function:

   image.DNAbin(alignment[,1:100])



   Using what you have already learned about R visualise the alignment for 5 samples of your choice between positions 12446 and 13161

8. To plot the relationships between the isolates we need to generate a phylogenetic tree. We are going to do this using the RAxML program which uses a Maximum Likelihood approach to phylogenetic inference. RAxML is not part of R but the ips package we have already loaded includes what is known as a wrapper function - we provide it with the options we wish to use with RAxML and it will run the program for us before loading the output into R for us. To start with we are going to construct a mid-rooted tree, which should look something like the one on the next page



9. A phylogenetic tree describes the relationship between the isolates visually. Before we can plot the tree we first have to calculate the relationships between each isolate. At the simplest level this will compare all pairwise combination of samples, identify the differences and count how many there are. The tree will then be plotted so that the most similar samples cluster together. More advanced models will take into account evolutionary processes, such as the fact that guanine is more likely to mutate into adenine than thymine. We will be using the Generalised Time Reversible (GTR) model with a gamma distribution.

   RAxML will use this model and calculate the relationship between each of the samples. It will then generate a mid-rooted tree and load the result into R for us. The command for this is:

   myalignment.rax.gtr <- raxml(alignment,
   m="GTRGAMMA",
   k=TRUE,
   f="a",
   p=1234,
   x=2345,
   N=100,
   file="ray2000",
   exec="raxmlHPC-PTHREADS-SSE3",
   threads=1)

10. This will save the results into a variable called myalignment.rax.gtr. Within this variable we have actually stored multiple trees. This is because we used a process called bootstrapping to determine the confidence of the branches on the tree. Bootstrapping work through a sampling with replacement process. We start with our original dataset and then generate a new one of equal size by randomly picking samples from our original dataset. Importantly we may pick the same sample multiple times. After plotting each of the bootstrapped trees we can then assess how frequently the same pattern of branching is observed.

    For now we are going to plot only the best tree, the following four commands each generate slightly different plots thanks to the use of the additional parameters after the + symbol

    ggtree(myalignment.rax.gtr$bestTree)
    ggtree(myalignment.rax.gtr$bestTree)+geom_treescale()
    ggtree(myalignment.rax.gtr$bestTree)+geom_treescale(width=0.01)
    ggtree(myalignment.rax.gtr$bestTree)+theme_tree2()

    The scale bar you have added to some of the plots displays the mean number of nucleotide substitutions per base. What is the mean number of substitutions between the samples at the base of your tree and at the far tip?

11. It is possible to plot our tree using a variety of different layouts by adding layout="" to our command, for example

    ggtree(myalignment.rax.gtr$bestTree, layout="fan")

    The possible layout options are: "rectangular", "slanted", "fan", "circular", "radial", "unrooted", "equal_angle" and "daylight". Try them and see what effect they have on the plot.

12. So far we have only plotted the tree itself, so it is impossible to tell which sample is which. To add some labels in add

    +geom_tiplab()

    to the end of the command you used for plotting. Note that for some of the layout options it will be better to use +geom_tiplab2() as this allows for the labels to be rotated.

    You may find that your labels extend off of the edge of the plot. This is because R scales the image portions of a plot automatically but uses a default font size for text. You will find similar

quirks with all programming languages if you choose to study them further. One quick way to address this is by extending the x-axis beyond the automaticaaly set value using:

+ggplot2::xlim(0, 0.04)

As always there are multiple ways of solving this problem and you need to ensure the solution is appropriate to your situation. We could have made the text smaller, the entire plot larger or extend the margin size of the plot. Each of these would have worked but could effect the plot in different ways if we were to change it later. For example if we added a new sample that was distantly related to the tree we may need to extend the x-axis even further to fit it and the labels into the new plot.

13. One of the useful extras that the ggplot2 package adds to R is the ability to save a plotting command into a variable, for example:

    plot1= ggtree(myalignment.rax.gtr$bestTree)

    Afterwards you could produce the plot by simply typing

    plot1

    You can also add additional features using the + command

    plot1=plot1+geom_tiplab()

14. The tree you have plotted was automatically rooted at the mid-point by RAxML based on the greatest tip-to-tip distance it calculated. Sometimes, however, we wish to define a specific root point based on prior knowledge (or beliefs) about how our samples are related to one another. Often this will be because we have included an outgroup sample that is distantly related to everything else. Here we are going to define our outgroup as our oldest samples, which all originated from the 1976 Ebola epidemic. To define the outgroup you will need to add the following line to you raxml command after ‘threads=2’ (don’t forget to add a , after threads=1)

    outgroup=c("AY142960.1","KC242791.1","AF499101.1","KC242801.1","AF272001.1","EU224440.2","AF086833.2")

Run the rxaml function and save the new results into a variable called myalignment.rax.rooted

15. Plot the new tree, how does it differ compared to the mid-point rooted tree?

16. Save the new plotting command into a variable of your choice and then plot the two trees sides by side using grid.arrange. The generic command for this is:

    grid.arrange(first_plot, second_plot)

17. We're now going to explore some of the additional options for plotting trees. As you may have already noticed with ggplot the command to plot a figure is built up of individual sections joined by + with each section adding a specific set of formatting or plot element. Using that knowledge test out these additional options in different combinations:

    +theme_tree()
    +theme_tree2()
    +geom_tiplab()
    +geom_nodepoint()

18. We discussed earlier that our trees were constructed using a process called bootstrapping. This works by taking all of the data, randomly picking samples and then constructing a new tree. We repeat this process a large number of times (we used 100 bootstraps) and compare all of them to determine if the topology (shape) of the tree is consistent.

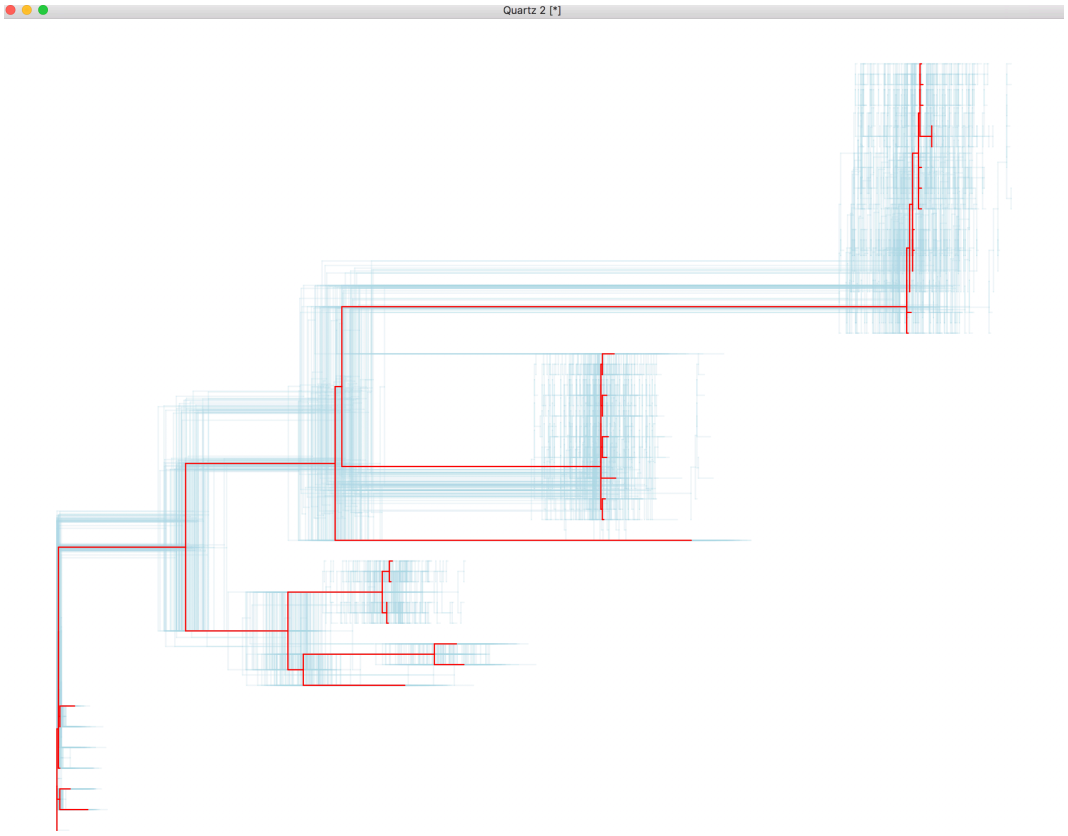
    Until now we have only been plotting the best tree but what if we wanted to investigate the individual trees generated by the bootstrapping? Luckily that data is also stored within myalignment.rax.gtr within $bootstrap. So we can plot the seventh tree using:

    ggtree(myalignment.rax.gtr$bootstrap[7])+theme_tree()

    That, however, is only one of one hundred. We could instead plot all 100 side by side using:

    ggtree(myalignment.rax.gtr$bootstrap)+facet_wrap(~.id,ncol=10)

    This take a minute or two to plot due to the large number of replicates. While we could use this plot to investigate the different trees it is a rather cumbersome approach. A better way would be to plot all of the trees in the background and then overlay the best tree as shown on the next page.



19. Work through the following commands to recreate a figure similar to the one above.

ggtree(myalignment.rax.gtr$bootstrap)+theme_tree()

ggtree(myalignment.rax.gtr$bootstrap, alpha=0.2)+theme_tree()

ggtree(myalignment.rax.gtr$bootstrap, alpha=0.2, color="gray")+theme_tree()

besttree=fortify(myalignment.rax.gtr$bestTree)

ggtree(myalignment.rax.gtr$bootstrap, alpha=0.2, color="gray",layout="rectangular")
+geom_tree(data=besttree, color="red")

Using what you have already learned add the sample labels to the tree.

20. The most common way of displaying bootstrap information is to present the value next to a branch if it is above a minimum threshold. For example if we see a value of 78 it means that we observed the same branches in 78% of out bootstrapped trees. For out data we can plot this using the following command:

    ggtree(myalignment.rax.gtr$bipartitions)+theme_tree2()+geom_tippoint()
    +geom_text2(aes(subset=(!isTip &
    as.numeric(label)>75),label=label),nudge_x=-0.003)+geom_tiplab()+ggplot2::xlim(0, 0.04)

    which will only display the bootstrap value if it is in excess of our threshold of 75.

    If you chance the threshold to 0 to show all bootstrap values you may notice that some are missing. Adjust your xlim values so that they show up on screen.

21. Now that you have plotted a variety of trees use them to identify the position of the two sequences that you mapped.

22. The reads that you mapped during this exercise came from a sample that was collected during the recent 2014 Ebola outbreak, however, our two references came from 2014 (ref1) and 1976 (ref2). Based on this knowledge do you think the samples are positioned correctly in the tree? Take a moment to discuss this with your neighbours and the demonstrators.

    The table on the next page provides some details about the origins of the samples, do any of the samples cluster in unusual positions compared to where or when they were collected?

Sample	Country of Origin	Sampling Date	Sample	Country of Origin	Sampling Date
AY142960	DRC	1976	KC242790	DRC	2007
KC242791	DRC	1976	HQ613402	DRC	2008
AF499101	DRC	1976	KC242784	DRC	2007
KC242801	DRC	1976	HQ613403	DRC	2007
AF272001	DRC	1976	KC242787	DRC	2007
EU224440	DRC	1976	KC242788	DRC	2007
AF086833	DRC	1976	KJ660347	GIN	2014
JQ352763	DRC	1995	KJ660346	GIN	2014
AY354458	DRC	1995	KJ660348	GIN	2014
KC242799	DRC	1995	KM034555	SLE	2014
KC242796	DRC	1995	KM034559	SLE	2014
KM519951	DRC	2014	KM034561	SLE	2014
KC242793	GABON	1996	KM233103	SLE	2014
KC242792	GABON	1994	KM233088	SLE	2014
KC242800	GABON	2002	KM034551	SLE	2014
KC242789	DRC	2007	KM233092	SLE	2014
KC242786	DRC	2007	KM233070	SLE	2014
KC242785	DRC	2007	KM233050	SLE	2014

Sampling dates and country of origin for each of the samples used in plotting your tree.