Pathogen Genomics Workshop

Joseph Crispell

07 Oct 2019

Introduction

Pathogens threaten the health of people and animals. Understanding pathogen transmission can help us understand how to control it.

Today we are going to be working with genomic data for the pathogen *Mycobacterium bovis*. *M. bovis* causes bovine tuberculosis in cattle and many other species. It costs Ireland millions to control. How can genomic data help?

We'll use some *M. bovis* genomic data sourced from infected cattle and wildlife to try and understand the role of wildlife. Wildlife species have been shown to harbour and transmit infection to cattle, we want to know if that is the case here in Ireland.

Learning objectives

After our workshop we hope that you have learnt about:

- Programming in R
- Loading and using a FASTA nucleotide file
- Constructing and plotting a phylogenetic tree
- Why github is really useful
- Using and building R packages

Step 1: Set your working directory

We'll be creating a few different plots over the course of todays workshop. As we progress, we would also recommend that you working along with us with your R script.

We'd recommend storing today's work (plots and your script) in a single folder. We'll start that process by creating a folder and setting it up as our working directory:

```
# Get the current date
today <- format(Sys.Date(), "%d-%m-%y")

# Create a new directory on our desktop to store todays outputs
directory <- file.path("~", "Desktop",
                       paste0("CRT_PathogenGenomicsWorkshop_", today), "")
dir.create(directory)

# Set your working directory
setwd(directory)
```

> QUESTION:
> 1. What is a working directory?

Step 2: Getting started

Next, we are going to install some R packages that we'll use throughout the workshop. The packages are `ape`, `phangorn` and `PathogenGenomicsWorkshopPackage`. The first two packages are commonly used for phylogenetic analyses in R.

The `PathogenGenomicsWorkshopPackage` is an R package that we have specifically developed for this course. It has a few functions that we'll use later on.

To install the R packages, you can use the following code:

```
# Install the 'ape' package
install.packages("ape", repos = "https://cloud.r-project.org")

# Install the 'phangorn' package
install.packages("phangorn", repos = "https://cloud.r-project.org")

# Install the 'devtools' package
install.packages("devtools", repos = "https://cloud.r-project.org")

# Install the 'pathogenGenomicsPackage' package
devtools::install_github("JosephCrispell/pathogenGenomicsWorkshop")
```

With the packages successfully installed, we can now load them:

```
# Load the required libraries
library(ape) # Handling sequences
library(phangorn) # # Building phylogeny
library(pathogenGenomicsWorkshop) # Our custom package
```

> QUESTION:
> 1. Why create/use R packages?

Step 3: Reading in the FASTA file

A FASTA file stores one or multiple nucleotide sequences. Our FASTA file stores the nucleotides present at a subset of genomic positions in `48` different *M. bovis* genomes. We can read in our *M. bovis* FASTA file with the following code:

```
# Read in the FASTA file
fastaFile <- system.file("extdata", "Mbovis.fasta",
                         package = "pathogenGenomicsWorkshop")
nucleotideAlignment <- read.dna(fastaFile, format = "fasta", as.character = TRUE)

# Convert the nucleotides to upper case
nucleotideAlignment <- toupper(nucleotideAlignment)
```

Notice by default nucleotides are stored in lower case, we've chosen to convert them to uppercase.

QUESTIONS:

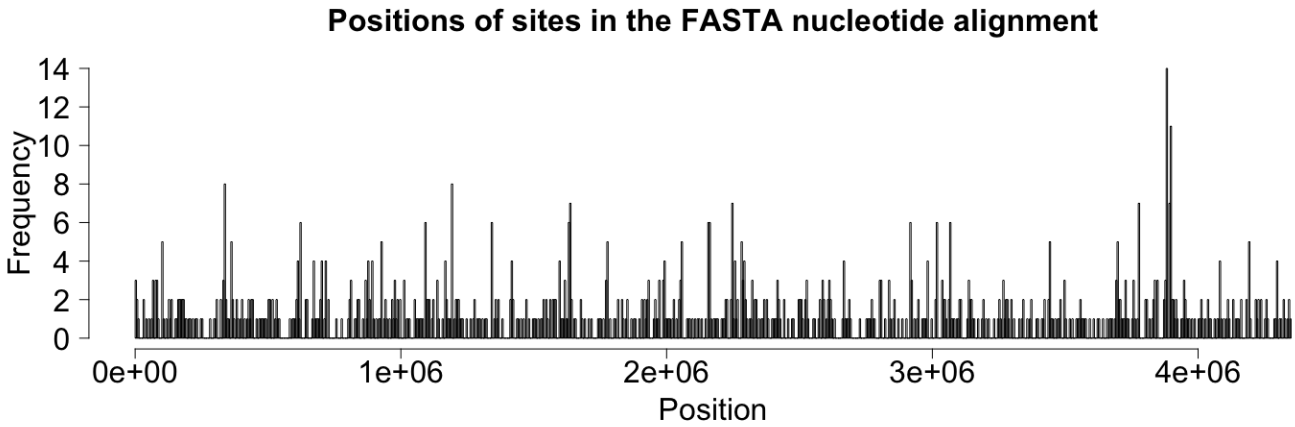
1. Can anyone tell me what class of variable we have stored the sequences in?
2. Do we have `48` sequences?
3. How many positions are in the FASTA file?

We also have a file that tells us which position on the *M. bovis* genome that each site in the FASTA file relates to. Let's read that in:

```
# Read in the genome positions
positionsFile <- system.file("extdata", "Mbovis_FASTApositions.txt",
                             package = "pathogenGenomicsWorkshop")
genomePositions <- read.table(positionsFile, header = TRUE)
```

EXERCISE:

1. Create the plot below

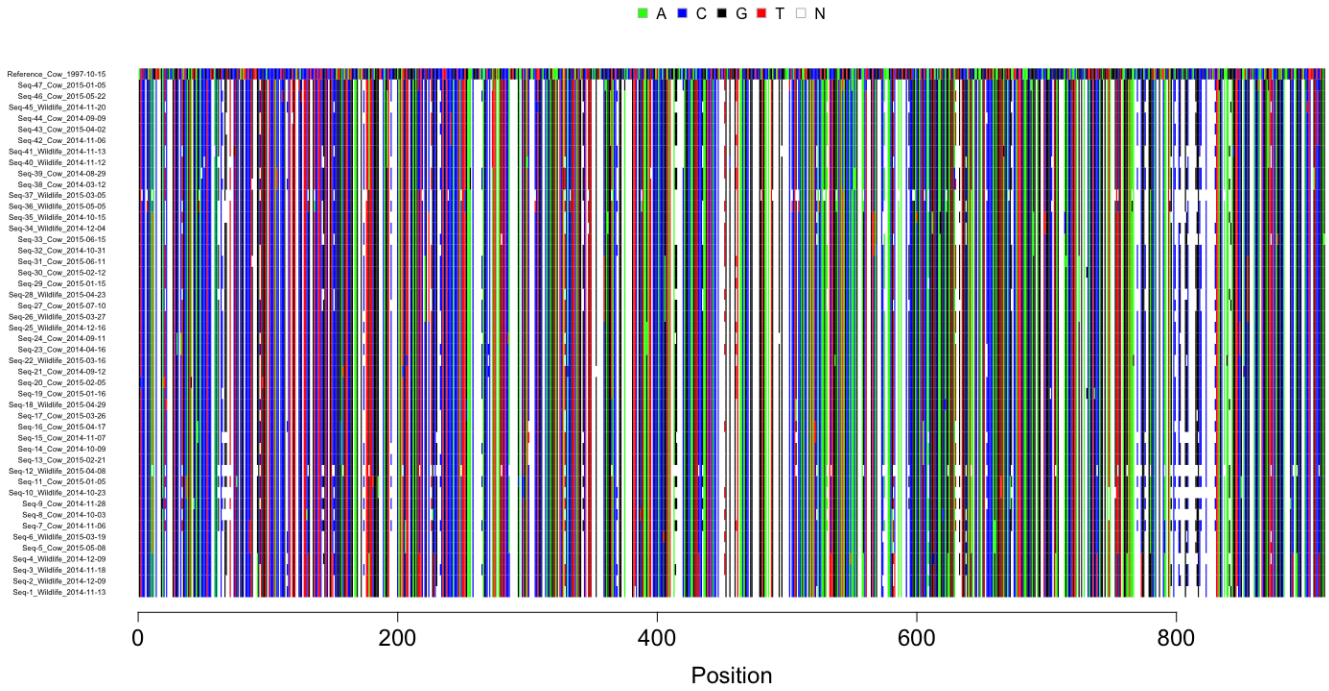


QUESTION:

1. Why might there be areas of the genome with more variants?

Let's also take a quick look at the FASTA file. The code below will create a plot that is saved as a PDF in your working directory:

```
plotFASTA(nucleotideAlignment,
          pdfFileName = paste0("FullNucleotideAlignment_", today, ".pdf"))
```



Step 4: Cleaning up the FASTA file

There are a lot of sites in the *M. bovis* alignment that aren't informative. We can clean up the alignment using the code below.

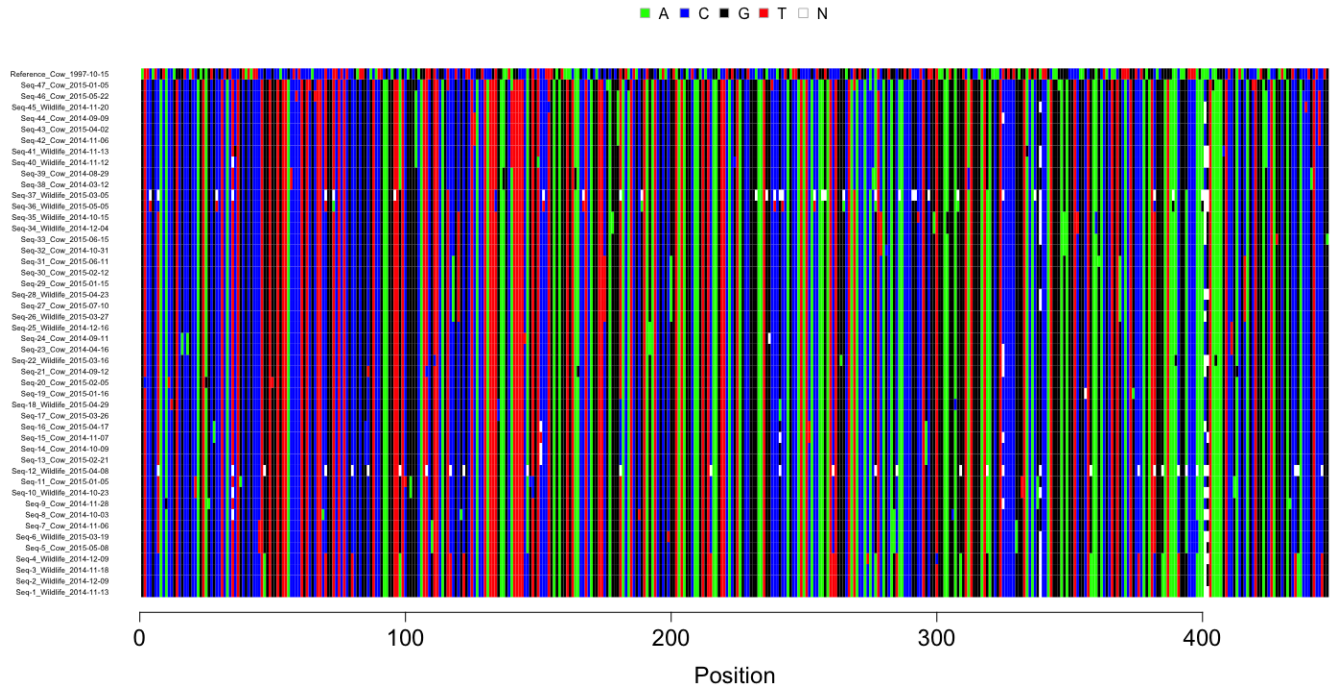
```
# Count the nucleotides at each site in the alignment
nucleotideCountsAtEachSite <- countNucleotidesAtEachSite(nucleotideAlignment)

# Identify the uninformative sites
uninformativeSites <- which(nucleotideCountsAtEachSite < 2)

# Create a new nucleotide alignment without the uninformative sites
nucleotideAlignmentInformative <- nucleotideAlignment[, -uninformativeSites]
informativeGenomePositions <- genomePositions[-uninformativeSites, ]
```

QUESTIONS:

1. What is an uninformative site?
2. What does line 5 in the above code block do?

Now, let's take another look at the alignment, how has it changed?

```
plotFASTA(nucleotideAlignmentInformative,
          pdfFileName = paste0("InformativeSitesAlignment_", today, ".pdf"))
```



QUESTIONS:

1. Can anyone guess what the nucleotide sequence at the top of the plot is?
2. Could removing uninformative sites have any effect?

Step 5: Extract the sequence metadata from the IDs

As you will have seen, the sequence labels in our alignment contain some information about our sequences. Let's extract these data and store them in a `data.frame`:

```
# Extract metadata from sequences
sequenceInfo <- getSequenceInfoFromNames(rownames(nucleotideAlignment))

# Take a quick look at the metadata
head(sequenceInfo)
```

```
##                          Name  Species SamplingDate
## 1 Seq-1_Wildlife_2014-11-13 Wildlife   2014-11-13
## 2 Seq-2_Wildlife_2014-12-09 Wildlife   2014-12-09
## 3 Seq-3_Wildlife_2014-11-18 Wildlife   2014-11-18
## 4 Seq-4_Wildlife_2014-12-09 Wildlife   2014-12-09
## 5       Seq-5_Cow_2015-05-08      Cow   2015-05-08
## 6 Seq-6_Wildlife_2015-03-19 Wildlife   2015-03-19
```

EXERCISE:

1. Calculate the number of samples sourced from wildlife and the number sourced from cattle

Step 6: Examine the quality of the nucleotide sequences

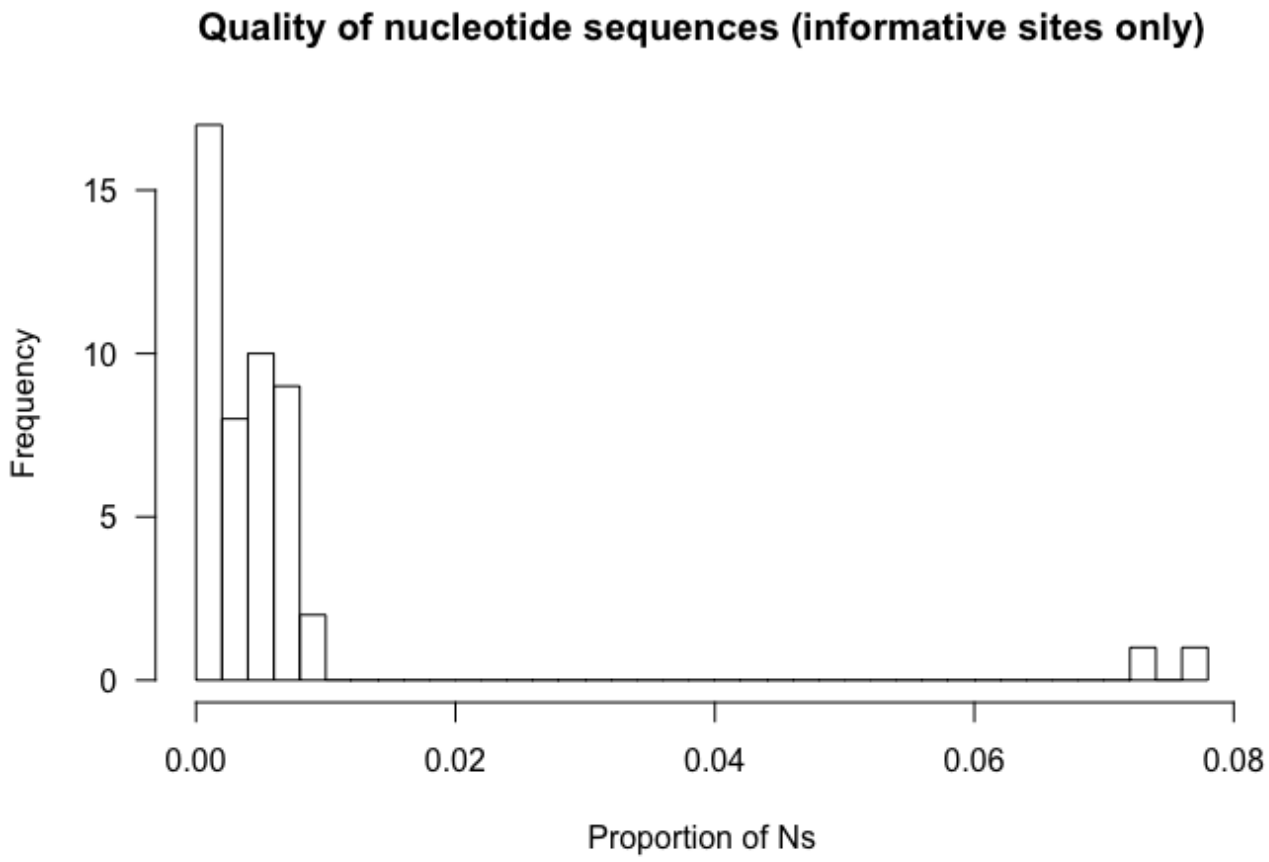
We don't have extensive data on the quality of our nucleotide sequences available but we can learn something about their quality from the nucleotide alignment. There are some `N`s in the alignment.

> QUESTION:
> 1. What do `N`s in a nucleotide alignment mean?

Let's calculate the proportion of nucleotides in each sequence that are `N`s:

```
# Calculate the proportion of Ns for each sequence
proportionNsInInformativeSites <-
  calculateProportionNsOfEachSequence(nucleotideAlignmentInformative)
```

> EXERCISE:
> 1. Create the plot below:



There are a couple of nucleotide sequences that don't have data for ~8% of the genome sites.

> QUESTION:
> 1. How might these differences in sequence quality impact our analyses?

Step 7: Build a phylogenetic tree

To build a phylogenetic tree we need to calculate the number of differences between each of our nucleotide sequences. We need to construct a genetic distance matrix:

```
# Build a genetic distance matrix
distances <- dist.dna(as.DNAbin(nucleotideAlignmentInformative), model = "raw")
```

Note that in the code above, we had to change the class (format) that we were storing our nucleotide alignment in.

Using the genetic distance matrix, we'll build an initial neighbour-joining phylogenetic tree:

```
# Build a quick initial phylogenetic tree
initialNJTree <- nj(distances)
```

The neighbour joining algorithm is a fast method to construct a phylogenetic tree but it isn't very robust. We are now going to construct a tree using the Maximum Likelihood algorithm. In addition, we are going to use bootstrapping to investigate the robusting of the phylogenetic tree structure.

> QUESTIONS:
> 1. Why is the Maximum Likelihood algorithm a more robust tree building algorithm?
> 2. How does bootstrapping work?

To prepare to build our Maximum Likelihood phylogeny, we'll construct a likelihood object using an initial tree and our nucleotide alignment:

```
# Convert the nucleotide sequences into the PHYDAT format
sequencesInPhyDatFormat <- phyDat(nucleotideAlignmentInformative, type = "DNA")

# Compute likelihood of the initial Neighbour Joining tree given sequences
likelihoodObject <- pml(initialNJTree, sequencesInPhyDatFormat)
```

With the likelihood object, we'll first run our maximum likelihood algorithm without bootstrapping:

```
# Run maximum likelihood
fittingOutput <- optim.pml(likelihoodObject,
                           optNni = TRUE, # Optimise topology
                           optInv = TRUE, # Optimise proportion of variable sites
                           optBf = TRUE, # Optimise the base frequencies
                           model = "HKY", # Substitution model
                           rearrangement = "NNI", # Nearest Neighbour Interchanges
                           control = pml.control(maxit = 100000)) # Max iterations
```

Lastly, now we'll take the output of our maximum likelihood analysis and feed it into a bootstrapping analysis:

```
# Build a bootstrapped maximum likelihood phylogeny
bootstrapResults <- bootstrap.pml(fittingOutput,
                                  bs = 100, # Number of bootstraps
                                  jumble = TRUE, # Jumble the order of the sequences
                                  control = pml.control(maxit = 100000)) # Max iters
```

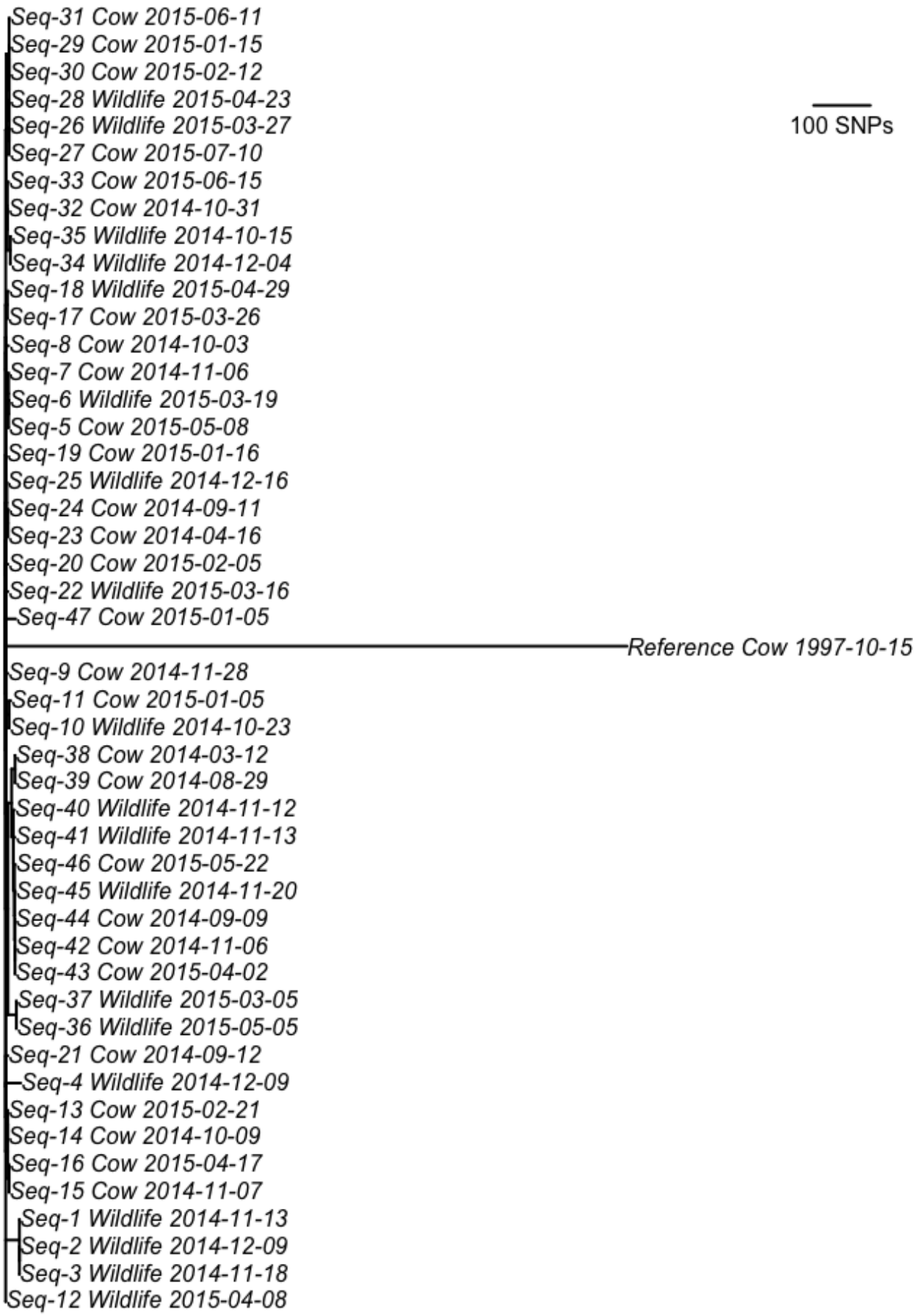
Step 8: Plotting the phylogenetic tree

Now that we have constructed and bootstrapped our maximum likelihood phylogenetic tree, let's take a look at it. First we'll need to extract the phylogeny from our bootstrapping output:

```
# Get phylogenetic tree with bootstrap values
# NOTE: plotBS() function returns tree with bootstrap values as node labels
mlTreeBS <- plotBS(fittingOutput$tree, bootstrapResults, type = "fan")
```

With the phylogeny stored as an object, we are going to create a simple plot:

```
# Convert the branch lengths to approximate SNPs
mlTreeBS$edge.length <- mlTreeBS$edge.length * ncol(nucleotideAlignmentInformative)

# Set the plotting margins
par(mar = c(0.1, 0.1, 0.1, 0.1))

# Root the phylogeny on the reference genome
mlTree <- root(mlTreeBS, outgroup="Reference_Cow_1997-10-15")

# Plot the phylogeny
plot.phylo(mlTreeBS, show.tip.label = TRUE, edge.width = 2, type = "phylogram")

# Add a scale
addSNPScale(position="topright", lineWidth = 2, size = 100)
```


Seq-31 Cow 2015-06-11
Seq-29 Cow 2015-01-15
Seq-30 Cow 2015-02-12
Seq-28 Wildlife 2015-04-23
Seq-26 Wildlife 2015-03-27
Seq-27 Cow 2015-07-10
Seq-33 Cow 2015-06-15
Seq-32 Cow 2014-10-31
Seq-35 Wildlife 2014-10-15
Seq-34 Wildlife 2014-12-04
Seq-18 Wildlife 2015-04-29
Seq-17 Cow 2015-03-26
Seq-8 Cow 2014-10-03
Seq-7 Cow 2014-11-06
Seq-6 Wildlife 2015-03-19
Seq-5 Cow 2015-05-08
Seq-19 Cow 2015-01-16
Seq-25 Wildlife 2014-12-16
Seq-24 Cow 2014-09-11
Seq-23 Cow 2014-04-16
Seq-20 Cow 2015-02-05
Seq-22 Wildlife 2015-03-16
Seq-47 Cow 2015-01-05
Reference Cow 1997-10-15
Seq-9 Cow 2014-11-28
Seq-11 Cow 2015-01-05
Seq-10 Wildlife 2014-10-23
Seq-38 Cow 2014-03-12
Seq-39 Cow 2014-08-29
Seq-40 Wildlife 2014-11-12
Seq-41 Wildlife 2014-11-13
Seq-46 Cow 2015-05-22
Seq-45 Wildlife 2014-11-20
Seq-44 Cow 2014-09-09
Seq-42 Cow 2014-11-06
Seq-43 Cow 2015-04-02
Seq-37 Wildlife 2015-03-05
Seq-36 Wildlife 2015-05-05
Seq-21 Cow 2014-09-12
Seq-4 Wildlife 2014-12-09
Seq-13 Cow 2015-02-21
Seq-14 Cow 2014-10-09
Seq-16 Cow 2015-04-17
Seq-15 Cow 2014-11-07
Seq-1 Wildlife 2014-11-13
Seq-2 Wildlife 2014-12-09
Seq-3 Wildlife 2014-11-18
Seq-12 Wildlife 2015-04-08
100 SNPs

QUESTIONS:
1. What does line 2 in the above code block do?
2. Why does the reference stick out so far?

Let's take a look at the boostrap support values:

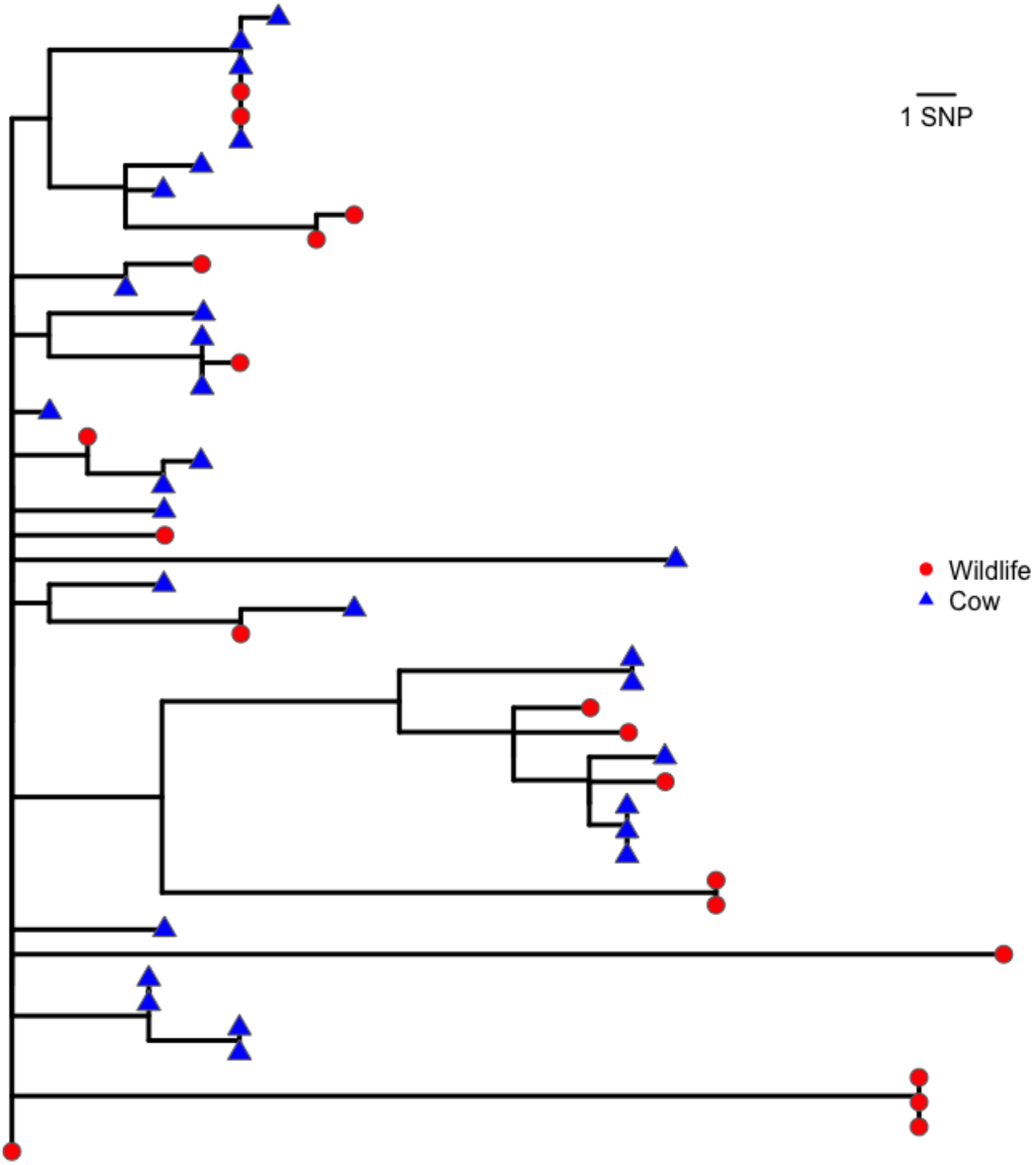
```
print(mlTreeBS$node.label)
```

```
##  [1] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
## [18] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
## [35] 100 100 100 100 100 100 100 100 100 100 100 100
```

QUESTION:
1. How much support for each node in the phylogeny is there?

Let's remove the reference sequence and take a closer look at the phylogenetic relationships:

```
# Remove the reference
mlTreeBSWithoutRef <- drop.tip(mlTreeBS, tip = "Reference_Cow_1997-10-15")

# Set the plotting margins
par(mar=c(0.1, 0.1, 0.1, 0.1))

# Plot the phylogeny - add in bootstrap values and species shapes
plot.phylo(mlTreeBSWithoutRef, show.tip.label = FALSE, edge.width = 3,
           type = "phylogram")

# Add node labels
isWildlife <- grepl(mlTreeBSWithoutRef$tip.label, pattern = "Wildlife")
tiplabels(pch = ifelse(isWildlife, 21, 24),
          bg = ifelse(isWildlife, "red", "blue"),
          col = "dimgrey", cex = 1.5)

# Add a scale
addSNPScale(position="topright", lineWidth = 2)

# Add a species legend
legend("right", legend = c("Wildlife", "Cow"), pch = c(19, 17),
       col = c("red", "blue"), bty = "n", xpd = TRUE)
```



QUESTIONS:

1. How similar are the *M. bovis* bacteria infected cattle and wildlife?
2. Does that have any implications for control?
3. What further analyses could we use on these data?

Step 9: Wrapping up

Today we have analyses nucleotide sequence data derived from whole genome sequence *M. bovis* data. *M. bovis* is an important bacterial pathogen.

Our samples were sourced from infected cattle and wildlife here in Ireland. Using these data, we were hoping to learn about what role wildlife are playing in Ireland's bovine tuberculosis problem.

Alongside the analyses of the *M. bovis* data, we've introduced aspects of programming in R, using github and creating and using an R package.

Some useful resources

To finish up, we would like to point out some helpful resources:

- Phylogenetic tree building (https://www.molecularecologist.com/2016/02/quick-and-dirty-tree-building-in-r/)
- Programming in R (https://www.tutorialspoint.com/r/index.htm)
- Forum to ask and answer questions (https://stackoverflow.com/questions/tagged/r)
- Building an R package (https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/)
- Getting started with github (https://guides.github.com/activities/hello-world/)
- Using R markdown (https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf)

Additional dataset

If you've raced through the analyses of the *M. bovis* data. We have added a *Zaire ebolavirus* (Ebola virus) nucleotide sequence alignment that you could work on. Ebola virus presents a massive risk to human health. The genomic data you'll examine is a subset of that sourced from infected people in West Africa more information here (https://www.nature.com/articles/nature22040).

There are some really nice visualisations of these data that can be found here (https://nextstrain.org/ebola). You can get started with the following code:

```
# Read in the FASTA file
fastaFile <- system.file("extdata", "ebola_parsed.fasta",
                         package = "pathogenGenomicsWorkshop")
nucleotideAlignment <- read.dna(fastaFile, format = "fasta", as.character = TRUE)

# Read in the genome positions
positionsFile <- system.file("extdata", "ebola_FASTApositions.txt",
                             package = "pathogenGenomicsWorkshop")
genomePositions <- read.table(positionsFile, header = TRUE)
```

Challenge

If you're having no trouble with the workshop content, take a look at the functions in our `pathogenGenomicsWorkshop` package.

Can you improve them?

You'll find the code for each of the functions here (https://github.com/JosephCrispell/pathogenGenomicsWorkshop/blob/master/R/pathogenGenomicsWorkshop.R). The `system.time()` function might be useful to calculate how long a function takes, and see if you can make it faster!