Reference set creation using *clstutils*

Noah Hoffman

April 25, 2023

Contents

1 Introduction

This vignette describes the use of functions in *clstutils* to create sets of reference sequences useful for performing phylogenetic-based taxonomic assignment. The primary inputs are an aligned set of sequences (in this case 16S rRNA), and annotation of taxonomic assignments.

```
> library(ape)
> library(lattice)
> library(clst)
> library(clstutils)
```

We will use data included in the package *clstutils* in the examples below. `seqs` is an object of class *DNAbin* representing a multiple sequence alignment, and `seqdat` is a *data.frame* containing taxonomic assignments of the sequences.

```
> data(seqs)
> data(seqdat)
```

2 Finding outliers

The example data contains sequences belonging to species in the genus *Enterococcus*.

```
> seqdat$i <- 1:nrow(seqdat)
> taxa <- split(seqdat, seqdat$tax_name)
> nseqs <- sapply(taxa, nrow)
> nseqs
```

```
   Enterococcus avium Enterococcus faecalis  Enterococcus faecium
                    7                    82                   111
```

2.1 Identifying outliers in a single taxon

Sequences obtained from public sources may not have correct taxonomic labels. When a sequence is incorrectly labeled as taxon A, we predict that it will have relatively large distances from other sequences that are correctly labeled as A. We will call these putatively mislabeled sequences *outliers*.

```
> Efaecium <- taxa[['Enterococcus faecium']]$i
```

Calculate a distance matrix using methods in *ape*.

```
> dmat <- ape::dist.dna(seqs[Efaecium,], pairwise.deletion=TRUE, as.matrix=TRUE, model='raw')
> summary(dmat[lower.tri(dmat)])
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.003358 0.005975 0.011041 0.013277 0.144397
```

The function `findOutliers` identifies a "most central" sequence S, and defines outliers as sequences with distances to S that exceed some threshold. This threshold can be provided explicitly:

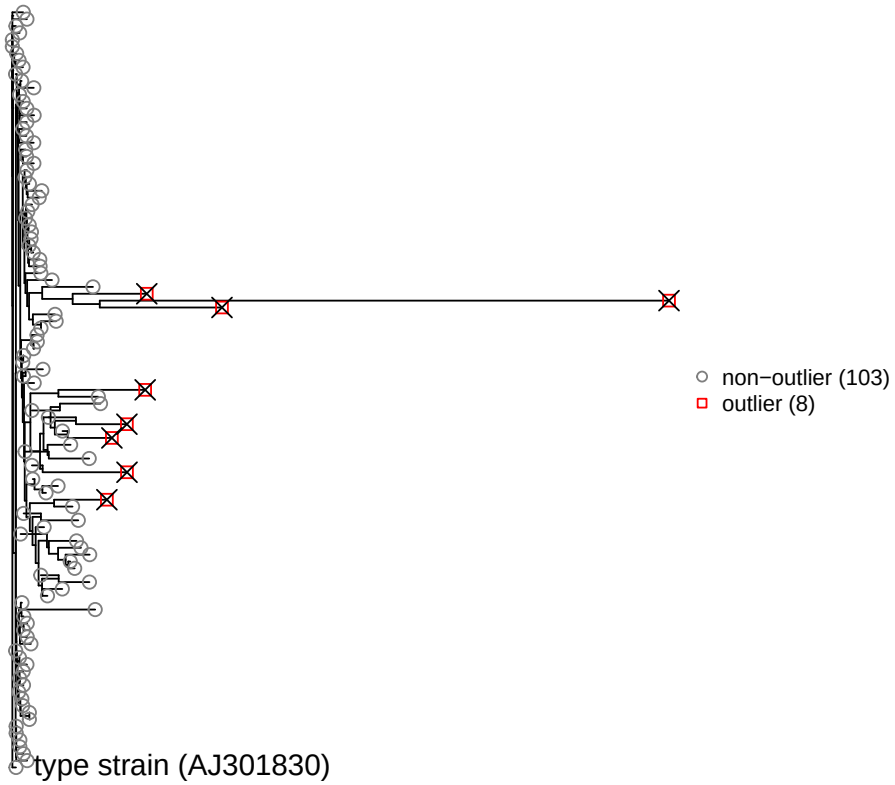
```
> outliers <- clstutils::findOutliers(dmat, cutoff=0.015)
> table(outliers)
```

```
outliers
FALSE  TRUE
  103     8
```

The threshold can also defined in terms of a quantile of all pairwise distances using the `quant` argument.

We can visualize the outliers on a phylogenetic tree (`clst::PrettyTree` extends `ape::plot.tree` to facilitate annotation). Note that the type strain for this

```
> with(seqdat[Efaecium,], {
+   prettyTree(nj(dmat), groups=ifelse(outliers,'outlier','non-outlier'),
+              X=outliers, labels=ifelse(isType,gettextf('type strain (%s)', accession),NA))
+ })
```



2.2 Outliers for multiple related taxa

First, generate a list of square distance matrices.

```
> dmats <- lapply(taxa, function(taxon) {
+   ape::dist.dna(seqs[taxon$i,], pairwise.deletion=TRUE, as.matrix=TRUE, model='raw')
+ })
```

Calculate outliers for each matrix. Here (as above) we are using a distance threshold of 1.5% from the "center-most" sequence (i.e., the one with the least sum of parwise distances to every other sequence).

```
> outliers <- sapply(dmats, findOutliers, cutoff=0.015)
```

Add results of outlier status to `seqdat`.

```
> seqdat$outlier <- FALSE
> for(x in outliers){
+   seqdat[match(names(x),seqdat$seqname),'outlier'] <- x
+ }
> with(seqdat, table(tax_name, outlier))
```

```
                       outlier
tax_name                FALSE TRUE
  Enterococcus avium        7    0
```

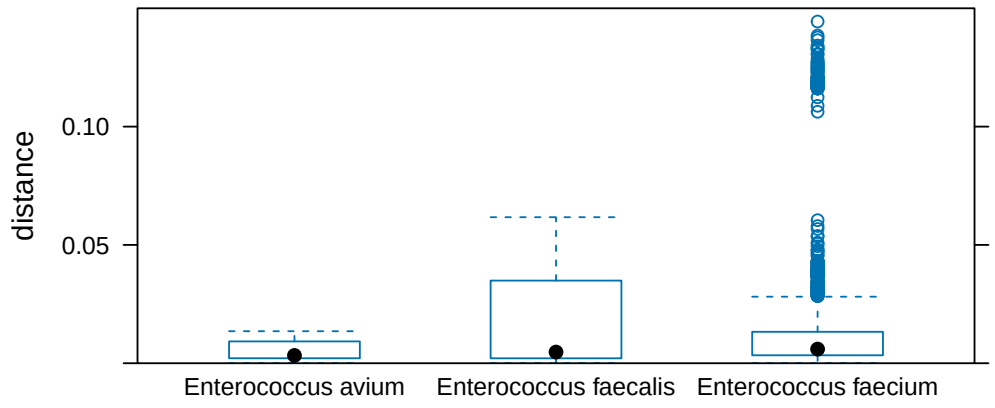
```
  Enterococcus faecalis    69   13
  Enterococcus faecium    103    8
```

It is instructive to visualize the effect of removing outliers on the distribution of within-species pairwise distances for each taxon. In the code fragment below, `omat` is a square matrix in which cells are TRUE if either margin is an outlier. We aggregate all of the pairwise distances in `dists`.

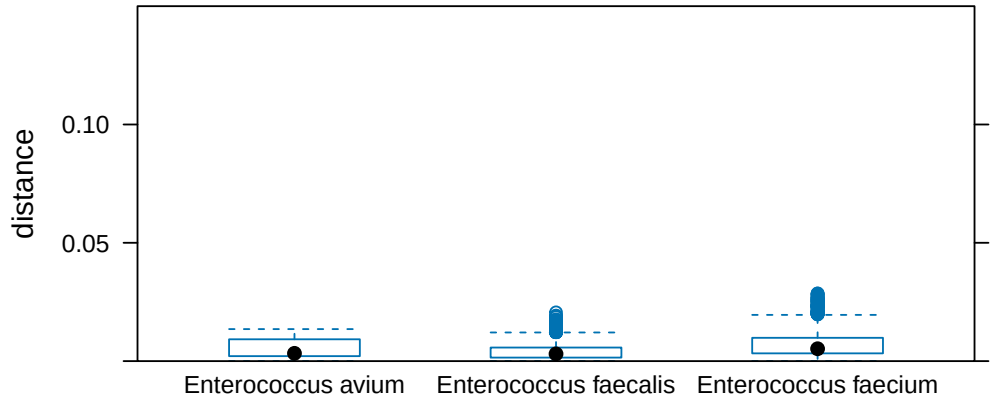
```
> lowerTriangle <- function(mat){mat[lower.tri(mat)]}
> dists <- do.call(rbind, lapply(names(dmats), function(tax_name){
+   dmat <- dmats[[tax_name]]
+   omat <- sapply(outliers[[tax_name]], function(i) {i | outliers[[tax_name]]})
+   data.frame(distance=lowerTriangle(dmat), outlier=lowerTriangle(omat))
+ }))
> dists$tax_name <- factor(rep(names(dmats), nseqs*(nseqs-1)/2))
> with(dists, table(tax_name, outlier))
```

```
                      outlier
tax_name               FALSE TRUE
  Enterococcus avium      21    0
  Enterococcus faecalis 2346  975
  Enterococcus faecium  5253  852
```

```
> plot(bwplot(distance ~ tax_name, data=dists, ylim=c(0,0.15)))
```

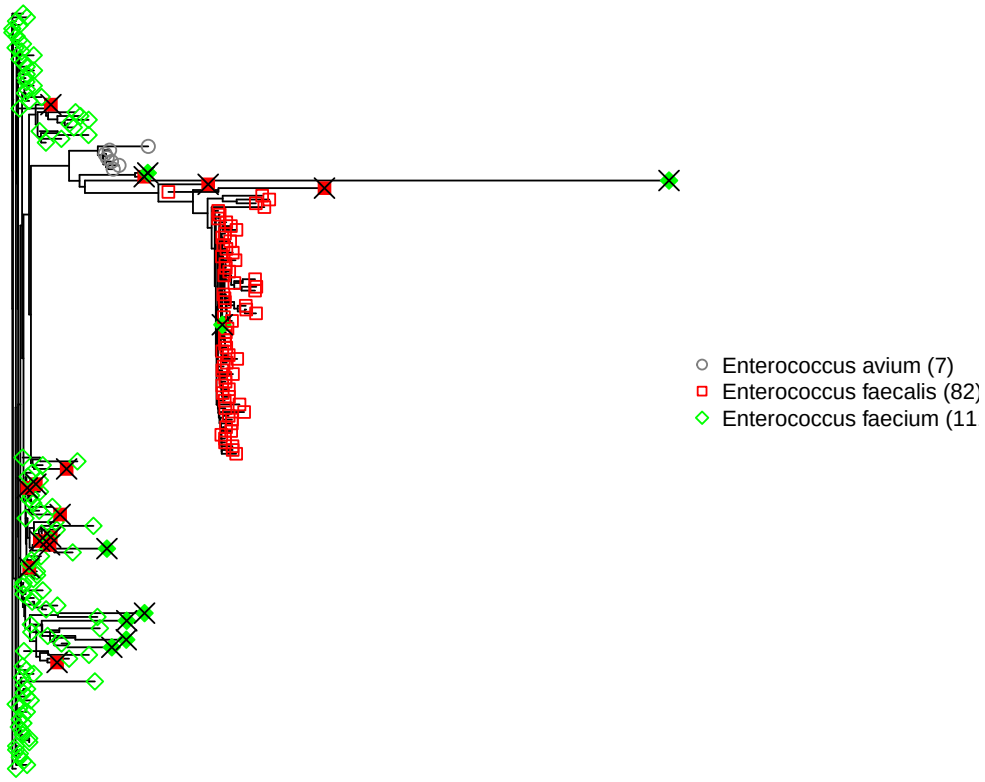



```
> plot(bwplot(distance ~ tax_name, data=subset(dists, !outlier), ylim=c(0,0.15)))
```



Finally, we can visualize the fact that many of the outliers are actually the result of labels being switched between taxa (that is, *E. faecium* sequenecs are labeled as *E. faecalis*) and vice versa. In the tree below, terminal nodes are identified according to the original species labels.

```
> with(seqdat, {
+   dmat <- ape::dist.dna(seqs, pairwise.deletion=TRUE, as.matrix=TRUE, model='raw')
+   clstutils::prettyTree(nj(dmat), groups=tax_name,
+                         ## type='unrooted',
+                         X=outlier, fill=outlier)
+ })
>
```

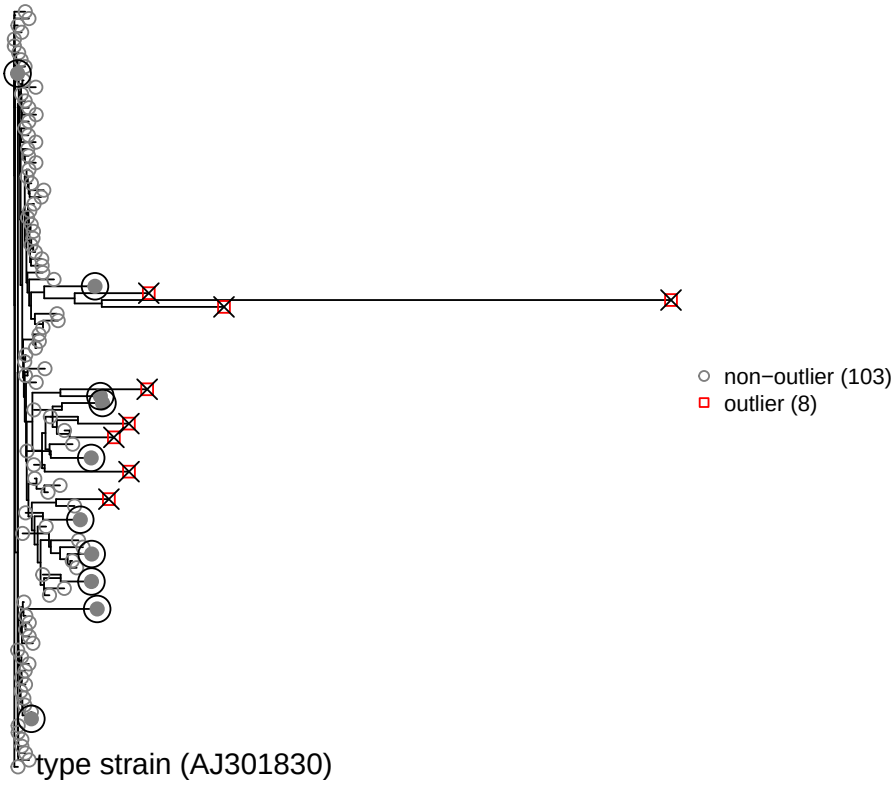


3 Selecting a diverse subset

Because we cannot use every available sequence in our reference tree, a sampling strategy is required. One strategy is to select a maximally diverse subset of sequences. The function `clstutils::maxDists` performs this operation. In addition, we can exclude sequences identified as outliers in the previous step (outlier identification is critical here, lest we select primarily outliers!). We can also optionally include the "centermost" sequence in the set, plus any type strains.

```
> with(seqdat[Efaecium,], {
+   selected <- clstutils::maxDists(dmat, idx=which(isType),
```

```
+                                    N=10, exclude=outlier, include.center=TRUE)
+   prettyTree(nj(dmat), groups=ifelse(outlier,'outlier','non-outlier'),
+              X=outlier,
+              O=selected, fill=selected,
+              labels=ifelse(isType,gettextf('type strain (%s)', accession),NA))
+ })
```



Here the selected sequences are identified with circled, filled glyphs.