Upsize your clustering with Clusterize

Erik S. Wright

May 1, 2024

Contents

1 Introduction to supersized clustering

You may have found yourself in a familiar predicament for many bioinformaticians: you have a lot of sequences and you need to downs*ize* before you can get going. You may also theor*ize* that this must be an easy problem to solve —given sequences, output clusters. But what can you util*ize* to solve this problem? This vignette will familiar*ize* you with the `Clusterize` function in the DECIPHER package. Clusterize will revolution*ize* all your clustering needs! Why `Clusterize`?:

- Scalability - `Clusterize` will linear*ize* the search space so that many sequences can be clustered in a reasonable amount of time.
- Simplicity - Although you can individual*ize* `Clusterize`, the defaults are straightforward and should meet most of your needs.
- Accuracy - `Clusterize` will maxim*ize* your ability to extract biologically meaningful results from your sequences.

This vignette will summar*ize* the use of `Clusterize` to cluster DNA, RNA, or protein sequences.

2 Getting started with Clusterize

To get started we need to load the DECIPHER package, which automatically mobil*ize* a few other required packages.

```
> library(DECIPHER)
```

There's no need to memor*ize* the inputs to `Clusterize`, because its help page can be accessed through:

```
> ? Clusterize
```

Note that, while it's easy to fantas*ize* about using `Clusterize`, if you only have a moderate number of **homologous** sequences (« 100k) then it's more accurate to use `TreeLine` with a distance matrix created from a multiple sequence alignment. This function provides hierarchical clustering (i.e., single-linkage, UPGMA, or complete-linkage) that is impossible to critic*ize* as inexact.

3 Optimize your inputs to Clusterize

`Clusterize` requires that you first digit*ize* your sequences by loading them into memory. For the purpose of this vignette, we will capital*ize* on the fact that DECIPHER already includes some built-in sets of sequences.

```
> # specify the path to your file of sequences:
> fas <- "<<path to training FASTA file>>"
> # OR use the example DNA sequences:
> fas <- system.file("extdata",
          "50S_ribosomal_protein_L2.fas",
          package="DECIPHER")
> # read the sequences into memory
> dna <- readDNAStringSet(fas)
> dna
DNAStringSet object of length 317:
      width seq                                               names
  [1]   819 ATGGCTTTAAAAAATTTTAATC...ATTTATTGTAAAAAAAAGAAAA Rickettsia prowaz...
  [2]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [3]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [4]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [5]   819 ATGGCTATCGTTAAATGTAAGC...CATCGTACGTCGTCGTGGTAAA Pasteurella multo...
  ...   ... ...
[313]   819 ATGGCAATTGTTAAATGTAAAC...TATCGTACGTCGCCGTACTAAA Pectobacterium at...
[314]   822 ATGCCTATTCAAAAATGCAAAC...TATTCGCGATCGTCGCGTCAAG Acinetobacter sp....
[315]   864 ATGGGCATTCGCGTTTACCGAC...GGGTCGCGGTGGTCGTCAGTCT Thermosynechococc...
[316]   831 ATGGCACTGAAGACATTCAATC...AAGCCGCCACAAGCGGAAGAAG Bradyrhizobium ja...
[317]   840 ATGGGCATTCGCAAATATCGAC...CAAGACGGCTTCCGGGCGAGGT Gloeobacter viola...
```

The `Clusterize` algorithm will general*ize* to nucleotide or protein sequences, so we must choose which we are going to use. Here, we hypothes*ize* that weaker similarities can be detected between proteins and, therefore, decide to use the translated coding (amino acid) sequences. If you wish to cluster at high similarity, you could also strateg*ize* that nucleotide sequences would be better because there would be more nucleotide than amino acid differences.

```
> aa <- translate(dna)
> aa
```

```
AAStringSet object of length 317:
      width seq                                               names
  [1]   273 MALKNFNPITPSLRELVQVDKT...STKGKKTRKNKRTSKFIVKKRK Rickettsia prowaz...
  [2]   274 MGIRKLKPTTPGQRHKVIGAFD...KGLKTRAPKKHSSKYIIERRKK Porphyromonas gin...
  [3]   274 MGIRKLKPTTPGQRHKVIGAFD...KGLKTRAPKKHSSKYIIERRKK Porphyromonas gin...
  [4]   274 MGIRKLKPTTPGQRHKVIGAFD...KGLKTRAPKKHSSKYIIERRKK Porphyromonas gin...
  [5]   273 MAIVKCKPTSAGRRHVVKIVNP...TKGKKTRHNKRTDKFIVRRRGK Pasteurella multo...
  ...   ... ...
[313]   273 MAIVKCKPTSPGRRHVVKVVNP...TKGKKTRSNKRTDKFIVRRRTK Pectobacterium at...
[314]   274 MPIQKCKPTSPGRRFVEKVVHS...KGYKTRTNKRTTKMIIRDRRVK Acinetobacter sp....
[315]   288 MGIRVYRPYTPGVRQKTVSDFA...SDALIVRRRKKSSKRGRGGRQS Thermosynechococc...
[316]   277 MALKTFNPTTPGQRQLVMVDRS...KKTRSNKSTNKFILLSRHKRKK Bradyrhizobium ja...
[317]   280 MGIRKYRPMTPGTRQRSGADFA...RKRRKPSSKFIIRRRKTASGRG Gloeobacter viola...
> seqs <- aa # could also cluster the nucleotides
> length(seqs)
[1] 317
```

Now you can choose how to parameter*ize* the function, with the main arguments being *myXStringSet* and *cutoff*. In this case, we will initial*ize cutoff* at `seq(0.5, 0, -0.1)` to cluster sequences from 50% to 100% similarity by 10%'s. It is important to recogn*ize* that *cutoff*s can be provided in *ascending* or *descending* order and, when *descending*, groups at each *cutoff* will be nested within the previous *cutoff*'s groups.

We must also choose whether to custom*ize* the calculation of distance. The defaults will penal*ize* gaps as single events, such that each consecutive set of gaps (i.e., insertion or deletion) is considered equivalent to one mismatch. If you want to standard*ize* the definition of distance to be the same as most other clustering programs then set: *penalizeGapLetterMatches* to `TRUE` (i.e., every gap position is a mismatch), *method* to `"shortest"`, *minCoverage* to `0`, and *includeTerminalGaps* to `TRUE`. It is possible to rational*ize* many different measures of distance – see the `DistanceMatrix` function for more information about alternative distance parameterizations.

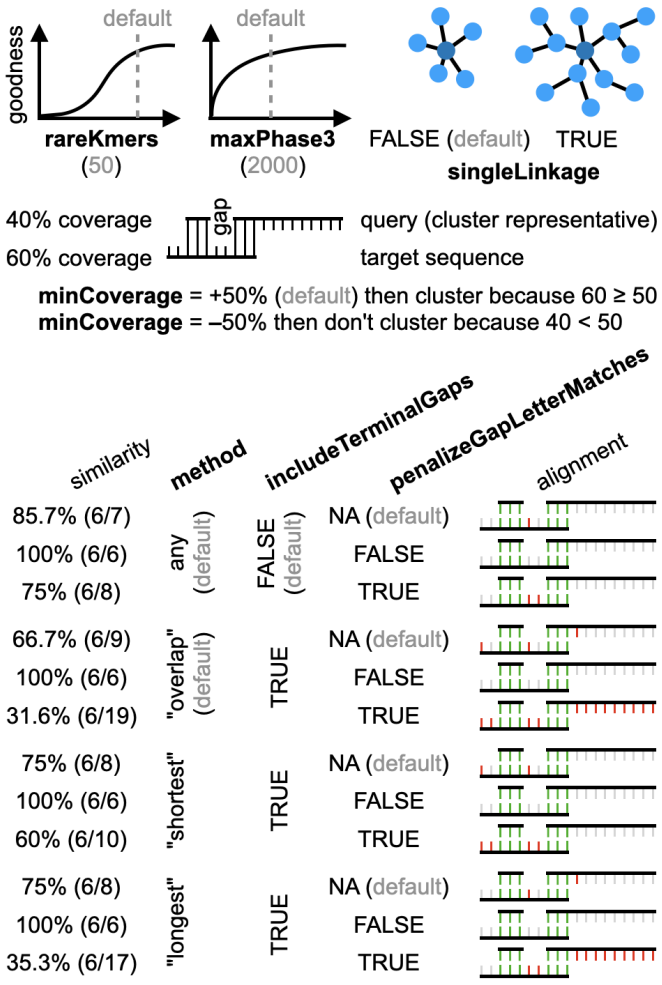

Figure 1: The most important parameters (in **bold**) to custom*ize* your use of `Clusterize`.

We can further personal*ize* the inputs as desired. The main function argument to emphas*ize* is *processors*, which controls whether the function is parallelized on multiple computer threads (if DECIPHER) was built with OpenMP enabled). Setting *processors* to a value greater than `1` will speed up clustering considerably, especially for large s*ize* clustering problems. Once we are ready, it's time to run `Clusterize` and wait for the output to material*ize*!

```
> clusters <- Clusterize(seqs, cutoff=seq(0.5, 0, -0.1), processors=1)
Partitioning sequences by 3-mer similarity:
================================================================================

Time difference of 0.06 secs

Sorting by relatedness within 35 groups:

iteration 34 of up to 34 (100.0% stability)

Time difference of 0.53 secs

Clustering sequences by 5-mer similarity:
================================================================================

Time difference of 0.15 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 3-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
> class(clusters)
[1] "data.frame"
> colnames(clusters)
[1] "cluster_0_5" "cluster_0_4" "cluster_0_3" "cluster_0_2" "cluster_0_1"
[6] "cluster_0"
> str(clusters)
'data.frame':        317 obs. of  6 variables:
 $ cluster_0_5: int  3 1 1 1 3 3 3 2 2 2 ...
 $ cluster_0_4: int  1 21 21 21 3 3 3 10 10 10 ...
 $ cluster_0_3: int  42 1 1 1 35 35 36 23 23 23 ...
 $ cluster_0_2: int  1 67 67 67 12 12 9 34 34 34 ...
 $ cluster_0_1: int  86 1 1 1 69 69 73 41 41 41 ...
 $ cluster_0  : int  2 102 102 102 25 25 20 59 59 59 ...
> apply(clusters, 2, max) # number of clusters per cutoff
cluster_0_5 cluster_0_4 cluster_0_3 cluster_0_2 cluster_0_1   cluster_0
          3          21          42          67          86         102
> apply(clusters, 2, function(x) which.max(table(x))) # max sizes
cluster_0_5 cluster_0_4 cluster_0_3 cluster_0_2 cluster_0_1   cluster_0
          3           5          30          22          54          45
```

Notice that `Clusterize` will character*ize* the clustering based on how many clustered pairs came from relatedness sorting versus rare k-mers, and `Clusterize` will predict the effectiveness of clustering. Depending on the input sequences, the percentage of clusters originating from relatedness sorting will equal*ize* with the number originating from rare k-mers, but more commonly clusters will originate from one source or the other. The clustering effectiveness formal*ize*s the concept of "inexact" clustering by approximating the fraction of possible sequence pairs

that were correctly clustered together. You can incentiv*ize* a higher clustering effectiveness by increasing *maxPhase3* at the expense of (proportionally) longer run times.

We can now real*ize* our objective of decreasing the number of sequences. Here, we will prioriti*ze* keeping only the longest diverse sequences.

```
> o <- order(clusters[[2]], width(seqs), decreasing=TRUE) # 40% cutoff
> o <- o[!duplicated(clusters[[2]])]
> aa[o]
AAStringSet object of length 21:
     width seq                                               names
 [1]   274 MGIRKLKPTTPGQRHKVIGAFDK...KGLKTRAPKKHSSKYIIERRKK Porphyromonas gin...
 [2]   274 MGIRKLKPTTPGQRHKVIGAFDK...KGLKTRAPKKHSSKYIIERRKK Porphyromonas gin...
 [3]   274 MAVRKLKPTTPGQRHKIIGTFEE...KGLKTRAPKKQSSKYIIERRKK Bacteroides theta...
 [4]   277 MGIKTYKPKTSSLRYKTTLSFDD...KGYKTRKKKRYSDKFIIKRRNK Borrelia burgdorf...
 [5]   280 MAIRKYKPTTPGRRQSSVSMFEE...NPNRYSNNMIVQRRRTNKSKKR Corynebacterium d...
 ...   ... ...
[17]   273 MAIVKCKPTSAGRRHVVKIVNPE...TKGKKTRHNKRTDKYIVRRRGK Haemophilus influ...
[18]   273 MAIVKCKPTSAGRRHVVKIVNPE...TKGKKTRHNKRTDKYIVRRRGK Haemophilus influ...
[19]   273 MAIVKCKPTSAGRRFVVKVVNQE...QTKGKKTRSNKRTDNMIVRRRK Pseudomonas aerug...
[20]   277 MALKHFNPITPGQRQLVIVDRSE...KKTRSNKATDKFIMRSRHQRKK Brucella suis VBI22
[21]   274 MAIVKCKPTSAGRRHVVKVVNAD...TKGYKTRSNKRTDKYIVRRRNK Vibrio cholerae PS15
> dna[o]
DNAStringSet object of length 21:
     width seq                                               names
 [1]   822 ATGGGAATACGTAAACTCAAGCC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
 [2]   822 ATGGGAATACGTAAACTCAAGCC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
 [3]   822 ATGGCAGTACGTAAATTAAAGCC...CATTATTGAGAGAAGAAAAAAG Bacteroides theta...
 [4]   831 ATGGGTATTAAGACTTATAAGCC...TATTATTAAAAGAAGAAATAAA Borrelia burgdorf...
 [5]   840 ATGGCTATTCGTAAGTACAAGCC...CACGAACAAGAGCAAGAAGCGC Corynebacterium d...
 ...   ... ...
[17]   819 ATGGCTATCGTTAAATGTAAGCC...TATCGTACGTCGTCGTGGCAAA Haemophilus influ...
[18]   819 ATGGCTATCGTTAAATGTAAGCC...TATCGTACGTCGTCGTGGCAAA Haemophilus influ...
[19]   819 ATGGCAATCGTTAAGTGCAAACC...CATGATCGTCCGCCGCCGCAAG Pseudomonas aerug...
[20]   831 ATGGCACTCAAGCATTTTAATCC...TTCGCGCCATCAGCGCAAGAAG Brucella suis VBI22
[21]   822 ATGGCTATTGTTAAATGTAAGCC...CATCGTACGTCGTCGTAATAAG Vibrio cholerae PS15
```

4 Visualize the output of Clusterize

We can scrutin*ize* the clusters by selecting them and looking at their multiple sequence alignment:

```
> t <- table(clusters[[1]]) # select the clusters at a cutoff
> t <- sort(t, decreasing=TRUE)
> head(t)
  3   1   2
218  58  41
> w <- which(clusters[[1]] == names(t[1]))
> AlignSeqs(seqs[w], verbose=FALSE)
AAStringSet object of length 218:
      width seq                                              names
```

```
  [1]   288 -MALKNFNPITPSLRELVQVDK...TRKNKRT-SKFIVKKRK----- Rickettsia prowaz...
  [2]   288 -MAIVKCKPTSAGRRHVVKIVN...TRHNKRT-DKFIVRRRGK---- Pasteurella multo...
  [3]   288 -MAIVKCKPTSAGRRHVVKIVN...TRHNKRT-DKFIVRRRGK---- Pasteurella multo...
  [4]   288 -MPLMKFKPTSPGRRSAVRVVT...TRKNKRT-QQFIVRDRRG---- Xanthomonas campe...
  [5]   288 -MPLMKFKPTSPGRRSAVRVVT...TRKNKRT-QQFIVRDRRG---- Xanthomonas citri...
  ...   ... ...
[214]   288 -MAFKHFNPTTPGQRQLVIVDR...TRSNKAT-DKFIMHTRHQRKK- Bartonella quinta...
[215]   288 -MAFKHFNPTTPGQRQLVIVDR...TRSNKAT-DKFIMHTRHQRKK- Bartonella quinta...
[216]   288 -MAIVKCKPTSPGRRHVVKVVN...TRSNKRT-DKFIVRRRTK---- Pectobacterium at...
[217]   288 -MPIQKCKPTSPGRRFVEKVVH...TRTNKRT-TKMIIRDRRVK--- Acinetobacter sp....
[218]   288 -MALKTFNPTTPGQRQLVMVDR...TRSNKST-NKFILLSRHKRKK- Bradyrhizobium ja...
```

It's possible to util*ize* the `heatmap` function to view the clustering results.

As can be seen in Figure 2, `Clusterize` will organ*ize* its clusters such that each new cluster is within the previous cluster when *cutoff* is provided in descending order. We can also see that sequences from the same species tend to cluster together, which is an alternative way to systemat*ize* sequences without clustering.

```
> aligned_seqs <- AlignSeqs(seqs, verbose=FALSE)
> d <- DistanceMatrix(aligned_seqs, verbose=FALSE)
> tree <- TreeLine(myDistMatrix=d, method="UPGMA", verbose=FALSE)
> heatmap(as.matrix(clusters), scale="column", Colv=NA, Rowv=tree)
```

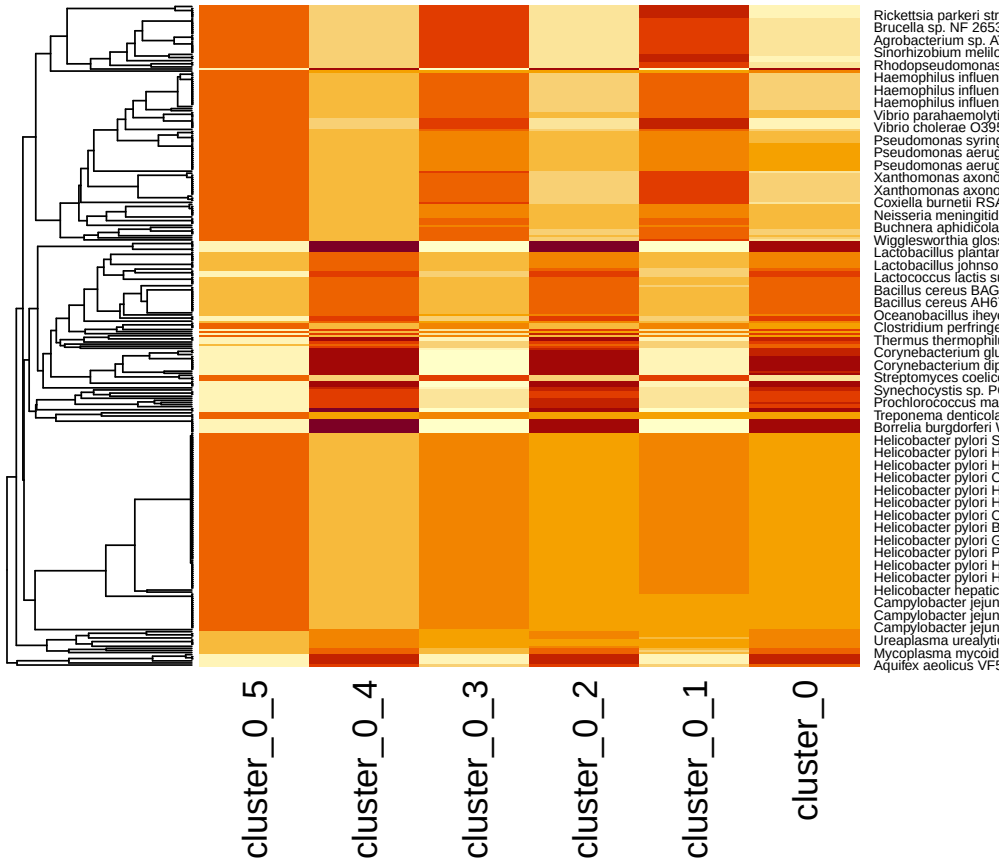

Figure 2: Visualization of the clustering.

5 Specialize clustering for your goals

The most common use of clustering is to categor*ize* sequences into groups sharing similarity above a threshold and pick one representative sequence per group. These settings empitom*ize* this typical user scenario:

```
> c1 <- Clusterize(dna, cutoff=0.2, invertCenters=TRUE, processors=1)
Partitioning sequences by 5-mer similarity:
================================================================================

Time difference of 0.12 secs

Sorting by relatedness within 34 groups:

iteration 25 of up to 56 (100.0% stability)

Time difference of 2.27 secs

Clustering sequences by 10-mer similarity:
================================================================================

Time difference of 0.42 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 5-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
> w <- which(c1 < 0 & !duplicated(c1))
> dna[w] # select cluster representatives (negative cluster numbers)
DNAStringSet object of length 78:
     width seq                                               names
 [1]   819 ATGGCTTTAAAAAATTTTAATCC...ATTTATTGTAAAAAAAAGAAAA Rickettsia prowaz...
 [2]   822 ATGGGAATACGTAAACTCAAGCC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
 [3]   837 GTGGGTATTAAGAAGTATAAACC...TGGTCGCCGTCCAGGCAAACAC Lactobacillus pla...
 [4]   825 ATGCCATTGATGAAGTTCAAACC...CATCGTCCGCGATCGTAGGGGC Xanthomonas axono...
 [5]   828 ATGGGTATTCGTAATTATCGGCC...GATTGTCCGCCGTCGCACCAAA Synechocystis sp....
 ...   ... ...
[74]   831 ATGGCATTTAAGCACTTTAATCC...TACGCGTCATCAGCGCAAGAAA Bartonella quinta...
[75]   843 ATGTTTAAGAAATATCGACCTGT...CGTGAAACGTCGAAGGAAGAAG Candidatus Protoc...
[76]   822 ATGCCTATTCAAAAATGCAAACC...TATTCGCGATCGTCGCGTCAAG Acinetobacter sp....
[77]   864 ATGGGCATTCGCGTTTACCGACC...GGGTCGCGGTGGTCGTCAGTCT Thermosynechococc...
[78]   840 ATGGGCATTCGCAAATATCGACC...CAAGACGGCTTCCGGGCGAGGT Gloeobacter viola...
```

By default, `Clusterize` will cluster sequences with linkage to the representative sequence in each group, but it is also possible to tell `Clusterize` to minim*ize* the number of clusters by establishing linkage to any sequence in the cluster (i.e., single-linkage). This is often how we conceptual*ize* natural groupings and, therefore, may better match alternative classification systems such as taxonomy:

```
> c2 <- Clusterize(dna, cutoff=0.2, singleLinkage=TRUE, processors=1)
Partitioning sequences by 5-mer similarity:
================================================================================

Time difference of 0.09 secs
```

```
Sorting by relatedness within 34 groups:

iteration 22 of up to 56 (100.0% stability)

Time difference of 1.96 secs

Clustering sequences by 10-mer similarity:
================================================================================

Time difference of 0.8 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 5-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
> max(abs(c1)) # center-linkage
[1] 78
> max(c2) # single-linkage (fewer clusters, but broader clusters)
[1] 76
```

It is possible to synthes*ize* a plot showing a cross tabulation of taxonomy and cluster number. We may ideal*ize* the clustering as matching taxonomic labels (3), but this is not exactly the case.

```
> genus <- sapply(strsplit(names(dna), " "), `[`, 1)
> t <- table(genus, c2[[1]])
> heatmap(sqrt(t), scale="none", Rowv=NA, col=hcl.colors(100))
```

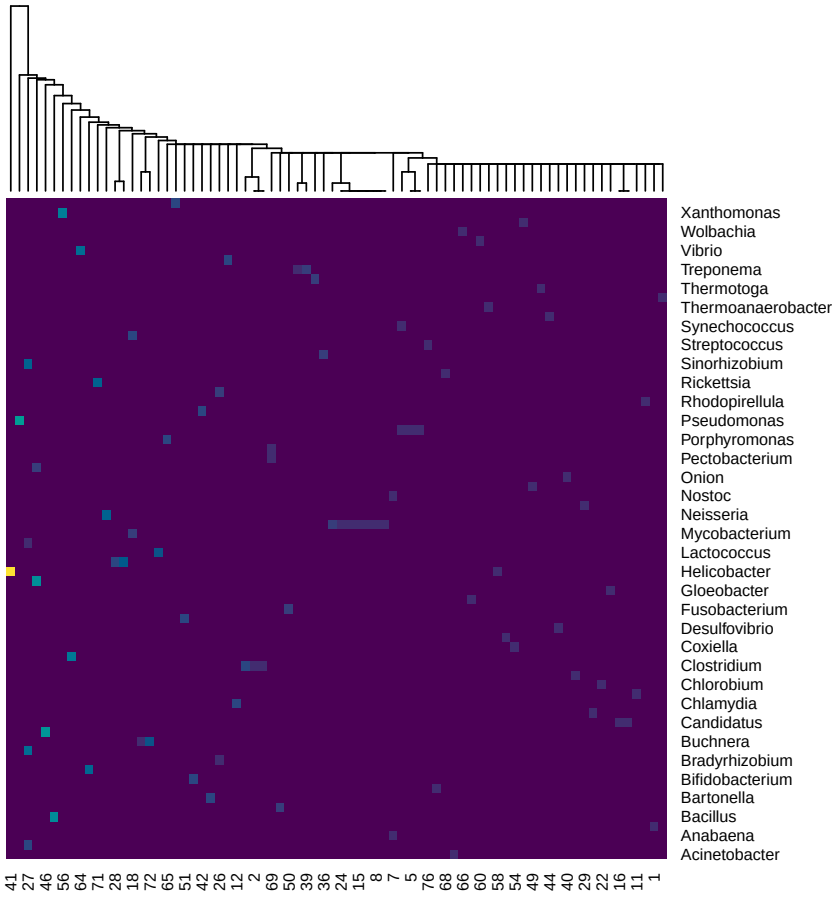

Figure 3: Another visualization of the clustering.

6 Resize to fit within less memory

What should you do if you have more sequences than you can cluster on your mids*ize* computer? If there are far fewer clusters than sequences (e.g., *cutoff* is high) then it is likely possible to res*ize* the clustering problem. This is accomplished by processing the sequences in batches that miniatur*ize* the memory footprint and are at least as large as the final number of clusters. The number of sequences processed per batch is critical to atom*ize* the problem appropriately while limiting redundant computations. Although not ideal from a speed perspective, the results will not jeopard*ize* accuracy relative to as if there was sufficient memory available to process all sequences in one batch.

```
> batchSize <- 2e2 # normally a large number (e.g., 1e6 or 1e7)
> o <- order(width(seqs), decreasing=TRUE) # process largest to smallest
> c3 <- integer(length(seqs)) # cluster numbers
> repeat {
         m <- which(c3 < 0) # existing cluster representatives
         m <- m[!duplicated(c3[m])] # remove redundant sequences
         if (length(m) >= batchSize)
                 stop("batchSize is too small")
         w <- head(c(m, o[c3[o] == 0L]), batchSize)
         if (!any(c3[w] == 0L)) {
                 if (any(c3[-w] == 0L))
                         stop("batchSize is too small")
                 break # done
         }
         m <- m[match(abs(c3[-w]), abs(c3[m]))]
         c3[w] <- Clusterize(seqs[w], cutoff=0.05, invertCenters=TRUE)[[1]]
         c3[-w] <- ifelse(is.na(c3[m]), 0L, abs(c3[m]))
 }
Partitioning sequences by 3-mer similarity:
================================================================================

Time difference of 0.02 secs

Sorting by relatedness within 4 groups:

iteration 1 of up to 29 (100.0% stability)

Time difference of 0.02 secs

Clustering sequences by 5-mer similarity:
================================================================================

Time difference of 0.13 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 3-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%

Partitioning sequences by 3-mer similarity:
================================================================================

Time difference of 0.03 secs
```

```
Sorting by relatedness within 97 groups:
Clustering sequences by 5-mer similarity:
================================================================================

Time difference of 0.22 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 3-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
> table(abs(c3)) # cluster sizes
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  1  1  1  1  1  1  2  1  1  2  1  3  1  1  1  1  1  7  1  1  1  3  1  1  1
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
 3  5  1  3  2  6  3  3  1  2  1  6  1  7  1  1  1  2  8  3 17  3  2  2  2  1
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 1  1  1  3  3 12  1 75  4  1  1 11  3  1  1  1  1  1  1  5  6  3  3  2  1  1
79 80 81 82 83 84 85 86 87 88 89 90 91
 1  1  1 17 13  6  3  1  1  1  1  1  1
```

7 Clustering both nucleotide strands

Sometimes the input sequences are present in different orientations and it is necessary to harmon*ize* the clusterings from both strands. Without trying to hyperbol*ize* how easy this is to do, here's an example of clustering both strands:

```
> # simulate half of strands having opposite orientation
> s <- sample(c(TRUE, FALSE), length(dna), replace=TRUE)
> dna[s] <- reverseComplement(dna[s])
> # cluster both strands at the same time
> clus <- Clusterize(c(dna, reverseComplement(dna)), cutoff=0.2, processors=1)
Partitioning sequences by 5-mer similarity:
================================================================================

Time difference of 0.23 secs

Sorting by relatedness within 142 groups:

iteration 28 of up to 50 (100.0% stability)

Time difference of 4.65 secs

Clustering sequences by 10-mer similarity:
================================================================================

Time difference of 1.21 secs

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 5-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
```

```
> clus <- match(clus[[1]], clus[[1]]) # renumber clusters ascending
> # if needed, reorient all clustered sequences to have the same orientation
> strand <- clus[seq_len(length(clus)/2)] >= clus[-seq_len(length(clus)/2)]
> dna[strand] <- reverseComplement(dna[strand])
> # renumber clusters across both strands and compare to original clustering
> clus <- pmin(clus[seq_len(length(clus)/2)], clus[-seq_len(length(clus)/2)])
> org <- match(abs(c1[[1]]), abs(c1[[1]])) # renumber original clustering
> mean(clus == org) # some differences expected due to algorithm stochasticity
[1] 0.9842271
> # verify the largest cluster is now back in the same orientation
> dna[clus == which.max(tabulate(clus))]
DNAStringSet object of length 75:
     width seq                                                names
 [1]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
 [2]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [3]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [4]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [5]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 ...   ... ...
[71]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[72]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[73]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[74]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[75]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
```

8 Finalize your use of Clusterize

Notably, `Clusterize` is a stochastic algorithm, meaning it will random*ize* which sequences are selected during pre-sorting. Even though the clusters will typically stabil*ize* with enough iterations, you can set the random number seed (before every run) to guarantee reproducibility of the clusters:

```
> set.seed(123) # initialize the random number generator
> clusters <- Clusterize(seqs, cutoff=0.1, processors=1)
Partitioning sequences by 3-mer similarity:
================================================================================

Time difference of 0.03 secs

Sorting by relatedness within 35 groups:

iteration 1 of up to 34 (100.0% stability)

Time difference of 0.02 secs

Clustering sequences by 5-mer similarity:
================================================================================

Time difference of 0.2 secs
```

```
Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 3-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
> set.seed(NULL) # reset the seed
```

Now you know how to util*ize* `Clusterize` to cluster sequences. To public*ize* your results for others to reproduce, make sure to provide your random number seed and version number:

- R version 4.4.0 RC (2024-04-16 r86468), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 22.04.4 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.20-bioc/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: BiocGenerics 0.51.0, Biostrings 2.73.0, DECIPHER 3.1.0, GenomeInfoDb 1.41.0, IRanges 2.39.0, S4Vectors 0.43.0, XVector 0.45.0
- Loaded via a namespace (and not attached): DBI 1.2.2, GenomeInfoDbData 1.2.12, R6 2.5.1, UCSC.utils 1.1.0, compiler 4.4.0, crayon 1.5.2, httr 1.4.7, jsonlite 1.8.8, tools 4.4.0, zlibbioc 1.51.0