Upsize your clustering with Clusterize

Erik S. Wright

April 25, 2023

Contents

1 Introduction

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

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
```

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
```

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 parameterizations.

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 6-mer similarity:

iteration 8 of up to 400 (100.0% coverage)

Time difference of 0.04 secs

Sorting by relatedness within 1 group:

iteration 51 of up to 51 (100.0% stability)

Time difference of 1.25 secs

Clustering sequences by 4-mer similarity:
================================================================================

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 4-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
Time difference of 0.18 secs
> 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  4 4 4 4 4 4 4 3 3 3 ...
 $ cluster_0_4: int  1 6 6 6 4 4 4 10 10 10 ...
 $ cluster_0_3: int  53 38 38 38 45 45 42 30 30 30 ...
 $ cluster_0_2: int  1 24 24 24 12 12 16 35 35 35 ...
 $ cluster_0_1: int  87 54 54 54 69 69 64 42 42 42 ...
 $ cluster_0  : int  2 45 45 45 24 24 32 57 57 57 ...
```

```
> 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
          4          27          53          71          87         102
```

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 priorit*ize* 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 27:
     width seq                                              names
 [1]   274 MAVRKLKPTTPGQRHKIIGTFEE...KGLKTRAPKKQSSKYIIERRKK Bacteroides sp. 1...
 [2]   274 MAVRKLKPTTPGQRHKIIGTFEE...KGLKTRAPKKQSSKYIIERRKK Bacteroides theta...
 [3]   276 MAIRKMKPITNGTRHMSRLVNDE...LGIKTRGRKTSDKFIVRRRNEK Fusobacterium nuc...
 [4]   280 MAIRKYKPTTPGRRASSVSMFTE...KPKRYSDDMIVRRRRANKNKKR Corynebacterium g...
 [5]   277 MGIKTYKPKTSSLRYKTTLSFDD...KGYKTRKKKRYSDKFIIKRRNK Borrelia burgdorf...
 ...   ... ...
[23]   276 MGIKKYNPTTNGRRNMTTNDFAE...LGFKTRKKNKASDKFIVRRRKK Bacillus thuringi...
[24]   278 MALKSFNPTTPSQRQLVIVSRAG...KRTRSNKSTDKFIMRSRHQRKK Sinorhizobium mel...
[25]   276 MSVIKCNPTSPGRRHVVKLVNGG...KGKRTRSNKRTDKFILCRRKKK Candidatus Blochm...
[26]   274 MAIVKCKPTSPGRRHVVKVVNAD...TKGFKTRKNKRTDKYIVRRRNK Vibrio parahaemol...
[27]   274 MAIVKCKPTSAGRRHVVKVVNAD...TKGYKTRSNKRTDKYIVRRRNK Vibrio cholerae 1...
> dna[o]
DNAStringSet object of length 27:
     width seq                                              names
 [1]   822 ATGGCAGTACGTAAATTAAAGCC...CATTATTGAGAGAAGAAAAAAG Bacteroides sp. 1...
 [2]   822 ATGGCAGTACGTAAATTAAAGCC...CATTATTGAGAGAAGAAAAAAG Bacteroides theta...
 [3]   828 ATGGCTATTAGAAAAATGAAACC...CGTAAGAAGAAGAAACGAAAAA Fusobacterium nuc...
 [4]   840 ATGGCTATTCGTAAGTACAAGCC...TGCTAACAAGAACAAGAAGCGC Corynebacterium g...
 [5]   831 ATGGGTATTAAGACTTATAAGCC...TATTATTAAAAGAAGAAATAAA Borrelia burgdorf...
 ...   ... ...
[23]   828 ATGGGAATCAAAAAGTATAATCC...CATCGTTCGTCGTCGTAAAAAA Bacillus thuringi...
[24]   834 ATGGCATTGAAAAGTTTCAATCC...CTCGCGTCACCAGCGCAAGAAG Sinorhizobium mel...
[25]   828 ATGTCTGTTATAAAATGTAATCC...TTTATGTCGTCGTAAGAAAAAA Candidatus Blochm...
[26]   822 ATGGCTATTGTTAAATGTAAGCC...CATCGTACGTCGTCGTAACAAG Vibrio parahaemol...
[27]   822 ATGGCTATTGTTAAATGTAAGCC...CATCGTACGTCGTCGTAATAAG Vibrio cholerae 1...
```

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)
  4   3   1   2 
153 105  47  12 
> w <- which(clusters[[1]] == names(t[1]))
> AlignSeqs(seqs[w], verbose=FALSE)
AAStringSet object of length 153:
      width seq                                               names               
  [1]   394 -MALKNFNPITPSLRELVQVDK...TR-KNKRTSKFIVKKRK----- Rickettsia prowaz...
  [2]   394 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [3]   394 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [4]   394 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [5]   394 -MAIVKCKPTSAGRRHVVKIVN...TR-HNKRTDKFIVRRRGK---- Pasteurella multo...
  ...   ... ...
[149]   394 -MAFKHFNPTTPGQRQLVIVDR...TR-SNKATDKFIMHTRHQRKK- Bartonella quinta...
[150]   394 -MAFKHFNPTTPGQRQLVIVDR...TR-SNKATDKFIMHTRHQRKK- Bartonella quinta...
[151]   394 -MAIVKCKPTSPGRRHVVKVVN...TR-SNKRTDKFIVRRRTK---- Pectobacterium at...
[152]   394 -MPIQKCKPTSPGRRFVEKVVH...TR-TNKRTTKMIIRDRRVK--- Acinetobacter sp....
[153]   394 -MALKTFNPTTPGQRQLVMVDR...TR-SNKSTNKFILLSRHKRKK- Bradyrhizobium ja...
```

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

As can be seen in Figure 1, `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)
```

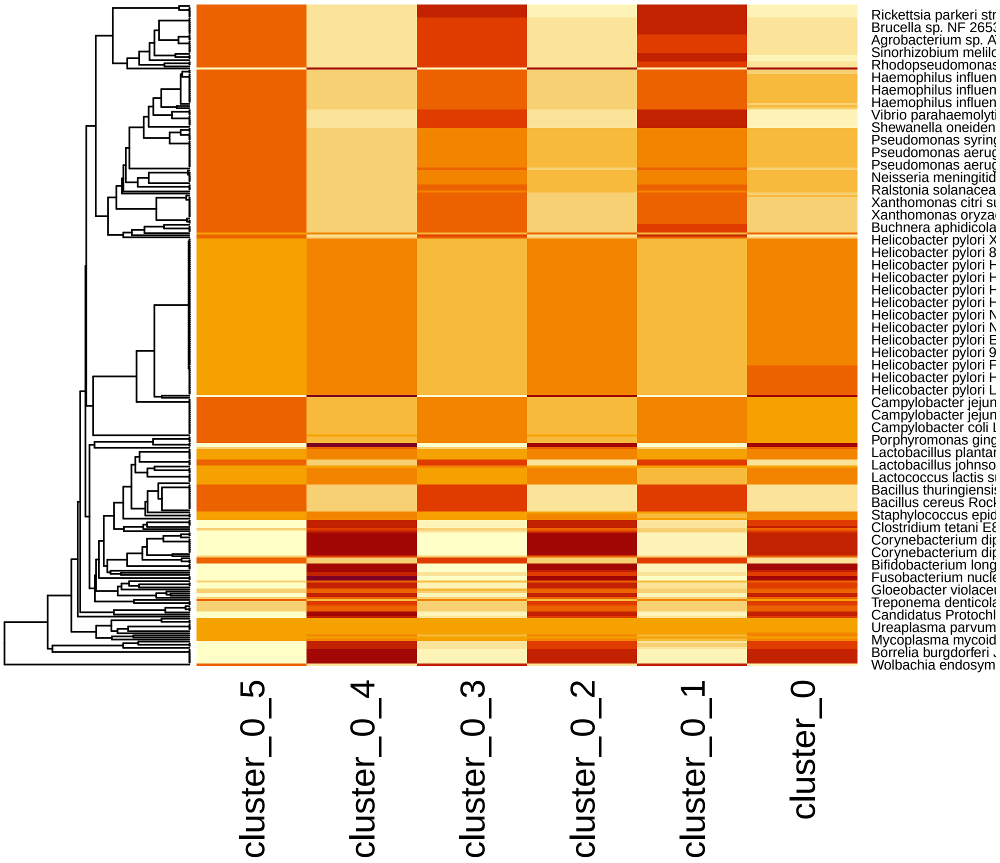

Figure 1: Visualization of the clustering.

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 9-mer similarity:

iteration 96 of up to 400 (100.0% coverage)

Time difference of 0.18 secs

Sorting by relatedness within 89 groups:

iteration 37 of up to 37 (100.0% stability)

Time difference of 0.49 secs

Clustering sequences by 9-mer similarity:
================================================================================

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 6-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
Time difference of 1.1 secs
> w <- which(c1 < 0 & !duplicated(c1))
> dna[w] # select cluster representatives (negative cluster numbers)
DNAStringSet object of length 78:
     width seq                                               names
 [1]   822 ATGGGAATACGTAAACTCAAGCC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
 [2]   825 ATGCCATTGATGAAGTTCAAACC...CATCGTCCGCGATCGTAGGGGC Xanthomonas campe...
 [3]   837 GTGGGTATTAAGAAGTATAAACC...TGGTCGCCGTCCAGGCAAACAC Lactobacillus pla...
 [4]   828 ATGGCAATTAAGAAGTATAAACC...CATTGTACGTCGTCGTAAAAAA Bacillus halodura...
 [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):

```
> c2 <- Clusterize(dna, cutoff=0.2, singleLinkage=TRUE, processors=1)
Partitioning sequences by 9-mer similarity:

iteration 96 of up to 400 (100.0% coverage)

Time difference of 0.17 secs
```

```
Sorting by relatedness within 89 groups:

iteration 37 of up to 37 (100.0% stability)

Time difference of 0.53 secs

Clustering sequences by 9-mer similarity:
================================================================================

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 6-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
Time difference of 1.76 secs
> 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 (2), 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))
```

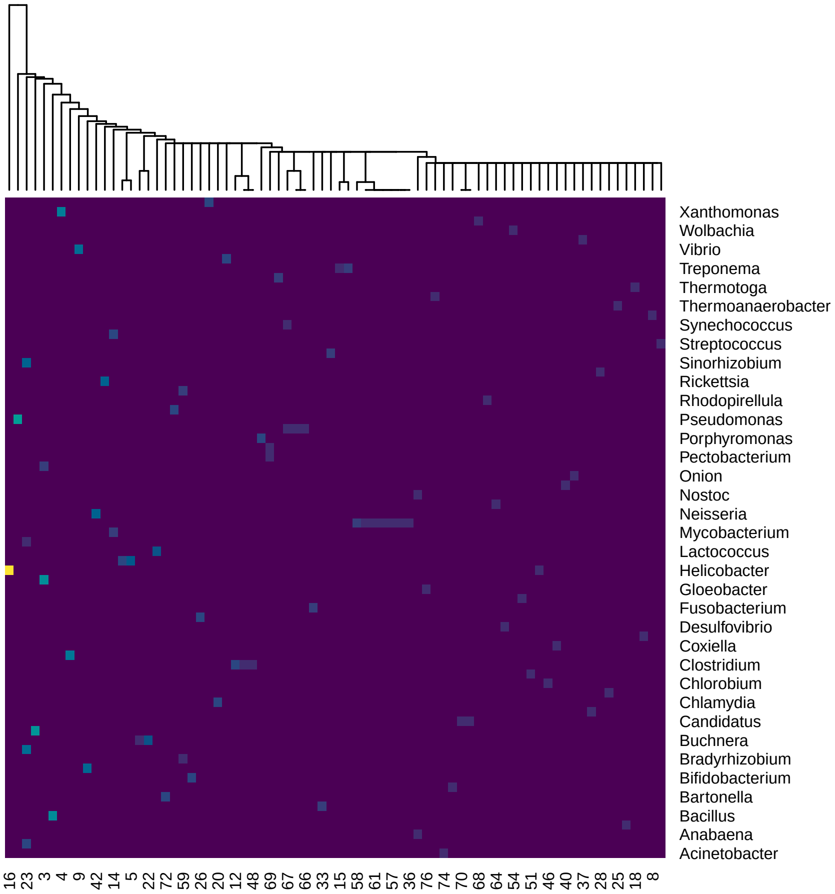

Figure 2: Another visualization of the clustering.

6 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 6-mer similarity:

iteration 8 of up to 400 (100.0% coverage)

Time difference of 0.03 secs

Sorting by relatedness within 1 group:

iteration 51 of up to 51 (100.0% stability)

Time difference of 0.76 secs

Clustering sequences by 4-mer similarity:
================================================================================

Clusters via relatedness sorting: 100% (0% exclusively)
Clusters via rare 4-mers: 100% (0% exclusively)
Estimated clustering effectiveness: 100%
Time difference of 0.08 secs
> set.seed(NULL) # reset the seed
```

Now you know how to util*ize* `Clusterize` to cluster sequences.

7 Session Information

All of the output in this vignette was produced under the following conditions:

- R version 4.3.0 RC (2023-04-13 r84269 ucrt), `x86_64-w64-mingw32`
- Running under: `Windows Server 2022 x64 (build 20348)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.46.0, Biostrings 2.68.0, DECIPHER 2.28.0, GenomeInfoDb 1.36.0, IRanges 2.34.0, RSQLite 2.3.1, S4Vectors 0.38.0, XVector 0.40.0
- Loaded via a namespace (and not attached): DBI 1.1.3, GenomeInfoDbData 1.2.10, RCurl 1.98-1.12, bit 4.0.5, bit64 4.0.5, bitops 1.0-7, blob 1.2.4, cachem 1.0.7, cli 3.6.1, compiler 4.3.0, crayon 1.5.2, fastmap 1.1.1, memoise 2.0.1, pkgconfig 2.0.3, rlang 1.1.0, tools 4.3.0, vctrs 0.6.2, zlibbioc 1.46.0