Upsize your clustering with Clusterize

Erik S. Wright

November 1, 2022

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 substitutions.

```
> 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 only required 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`.

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)
Ordering sequences by 4-mer similarity:

iteration 2 of up to 51 (100.0% stability)

Time difference of 0.44 secs

Clustering sequences by similarity:
================================================================================

Time difference of 0.4 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  52 37 37 37 44 44 41 27 27 27 ...
 $ cluster_0_2: int  1 24 24 24 12 12 16 37 37 37 ...
 $ cluster_0_1: int  87 54 54 54 69 69 64 40 40 40 ...
 $ cluster_0  : int  2 45 45 45 24 24 32 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
          4          26          52          70          87         102
```

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 26:
     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...
 ...   ... ...
[22]   277 MAIKKYKPSSNGRRGMTTSDFAE...EFKTRKQKNKSDKFIVRRRKNK Bacillus subtilis...
[23]   276 MGIKKYNPTTNGRRNMTTNDFAE...LGFKTRKKNKASDKFIVRRRKK Bacillus thuringi...
[24]   278 MALKSFNPTTPSQRQLVIVSRAG...KRTRSNKSTDKFIMRSRHQRKK Sinorhizobium mel...
[25]   276 MSVIKCNPTSPGRRHVVKLVNGG...KGKRTRSNKRTDKFILCRRKKK Candidatus Blochm...
[26]   274 MAIVKCKPTSAGRRHVVKVVNAD...TKGYKTRSNKRTDKYIVRRRNK Vibrio cholerae 1...
> dna[o]
DNAStringSet object of length 26:
     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...
 ...   ... ...
[22]   831 ATGGCGATTAAAAAGTATAAACC...CGTACGTCGTCGTAAAAATAAA Bacillus subtilis...
[23]   828 ATGGGAATCAAAAAGTATAATCC...CATCGTTCGTCGTCGTAAAAAA Bacillus thuringi...
[24]   834 ATGGCATTGAAAAGTTTCAATCC...CTCGCGTCACCAGCGCAAGAAG Sinorhizobium mel...
[25]   828 ATGTCTGTTATAAAATGTAATCC...TTTATGTCGTCGTAAGAAAAAA Candidatus Blochm...
[26]   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  45  14
> w <- which(clusters[[1]] == names(t[1]))
> AlignSeqs(seqs[w], verbose=FALSE)
AAStringSet object of length 153:
      width seq                                               names
  [1]   287 -MALKNFNPITPSLRELVQVDK...TR-KNKRTSKFIVKKRK----- Rickettsia prowaz...
  [2]   287 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [3]   287 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [4]   287 -MGIRKLKPTTPGQRHKVIGAF...TRAPKKHSSKYIIERRKK---- Porphyromonas gin...
  [5]   287 -MAIVKCKPTSAGRRHVVKIVN...TR-HNKRTDKFIVRRRGK---- Pasteurella multo...
  ...   ... ...
[149]   287 -MAFKHFNPTTPGQRQLVIVDR...TR-SNKATDKFIMHTRHQRKK- Bartonella quinta...
[150]   287 -MAFKHFNPTTPGQRQLVIVDR...TR-SNKATDKFIMHTRHQRKK- Bartonella quinta...
[151]   287 -MAIVKCKPTSPGRRHVVKVVN...TR-SNKRTDKFIVRRRTK---- Pectobacterium at...
[152]   287 -MPIQKCKPTSPGRRFVEKVVH...TR-TNKRTTKMIIRDRRVK--- Acinetobacter sp....
[153]   287 -MALKTFNPTTPGQRQLVMVDR...TR-SNKSTNKFILLSRHKRKK- Bradyrhizobium ja...
```

```
> heatmap(as.matrix(clusters), scale="column", Colv=NA)
```

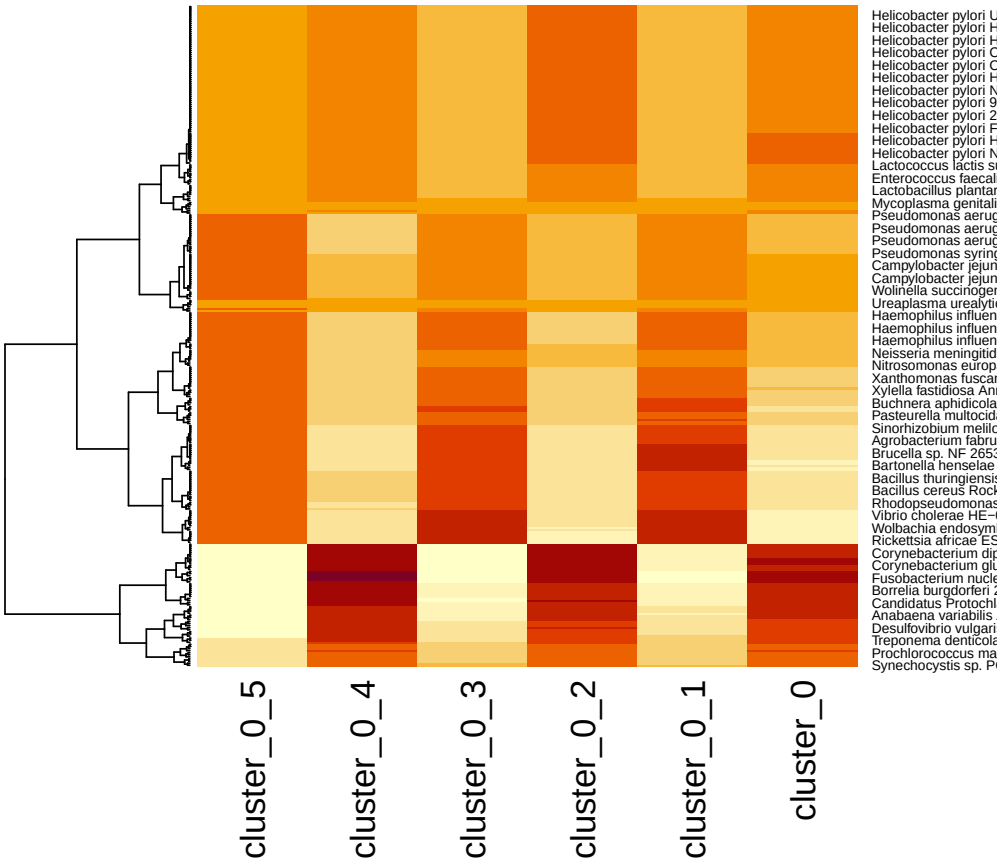

Figure 1: Visualization of the clustering.

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 another way to categor*ize* sequences without clustering.

5 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 to guarantee reproducibility of the clusters:

```
> set.seed(123) # initialize the random number generator
> clusters <- Clusterize(seqs, cutoff=seq(0.5, 0, -0.1))
Ordering sequences by 4-mer similarity:

iteration 2 of up to 51 (100.0% stability)

Time difference of 0.47 secs

Clustering sequences by similarity:
================================================================================

Time difference of 0.32 secs
> set.seed(NULL) # reset the seed
```

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

6 Session Information

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

- R version 4.2.1 (2022-06-23 ucrt), `x86_64-w64-mingw32`
- Running under: `Windows Server x64 (build 20348)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.44.0, Biostrings 2.66.0, DECIPHER 2.26.0, GenomeInfoDb 1.34.0, IRanges 2.32.0, RSQLite 2.2.18, S4Vectors 0.36.0, XVector 0.38.0
- Loaded via a namespace (and not attached): DBI 1.1.3, GenomeInfoDbData 1.2.9, RCurl 1.98-1.9, Rcpp 1.0.9, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.3, cachem 1.0.6, cli 3.4.1, compiler 4.2.1, crayon 1.5.2, fastmap 1.1.0, memoise 2.0.1, pkgconfig 2.0.3, rlang 1.0.6, tools 4.2.1, vctrs 0.5.0, zlibbioc 1.44.0