BioTools: Tools based on Biostrings (alignment, classification, database)

Michael Hahsler
Southern Methodist University

Anurag Nagar
Southern Methodist University

Abstract

Three are many stand-alone tools available for Bioinformatics. This package aims at using R and the Biostrings package as the common interface for several important tools for multiple sequence alignment (clustalw, kalign), classification (RDP), sequence retrieval (BLAST) as well as database driven sequence management for 16S rRNA.

Keywords: bioinformatics, Bioconductor, biostrings, sequence alignment, sequence classification, sequence management.

1. Introduction

There are many tools available for sequence alignment and classification. Some tools are: BAlibase (Smith and Waterman 1981), BLAST (Altschul, Gish, Miller, Myers, and Lipman 1990), T-Coffee (Notredame, Higgins, and Heringa 2000), MAFFT (Katoh, Misawa, Kuma, and Miyata 2002), MUSCLE (Edgar 2004b,a), Kalign (Lassmann and Sonnhammer 2006) and ClustalW2 and ClustalX2 (Larkin, Blackshields, Brown, Chenna, McGettigan, McWilliam, Valentin, Wallace, Wilm, Lopez, Thompson, Gibson, and Higgins 2007). Typically, these tools have a command-line interface and the input and output data is stored in files using various formats. Also the parameters supplied to the command-line interface are different. All this makes using and comparing several approaches time consuming and error prone. The R-based Bioconductor project (**?**) provides important infrastructure to handle and manipulate bioinformatics data. The **Biostrings** package in particular provides infrastructure for DNA, RNA and protein sequences as well as (multiple) alignments. Also algorithms for sequence alignment are included. However, for multiple sequence alignment and using BLAST the user needs to export the data into a file and then run the needed tool manually and re-import the results. Also, **Biostrings** stores sets of sequences in memory and does not directly support storing and querying classification information.

In **BioTools** we provide a simple interface to a growing set of popular tools. The tools are called directly from within R and no manual data export or import is needed. Currently we interface *clustalw, kalign, RDP* and *clustalw*. **BioTools** also provides database backed sequence management. Where large amounts of sequences and classification information can be stored and used for selective sequence retrieval.

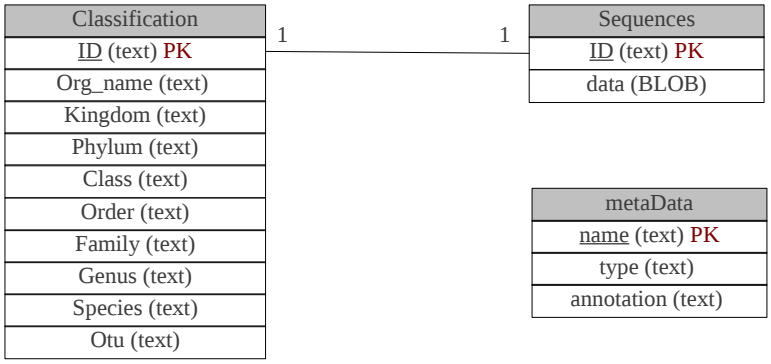

Figure 1: Entity Relationship diagram of GenDB

2. GenDB: Sequence storage an management

BioTools provides a databases (GenDB) which can be used for efficient storage and retrieval of genetic sequences. By default the light-weight SQLite database is used, but any other compatible database such as mySQL or Oracle can also be used. Figure 1 shows the basic table layout of a GenDB instance with a table containing classification information, a table containing the sequence information and a meta data table. Each sequence we will have an entry in the classification table and an corresponding entry in the sequence table. The tables are connected by a unique sequence ID as the primary key.

GenDB is easy to use. First, we load the library into the R environment.

```
R> library(BioTools)
```

To start we need to create an empty GenDB to store and organize sequences.

```
R> db<-createGenDB("example.sqlite")
R> db

Object of class GenDB with 0 sequences
DB File: example.sqlite
Tables: sequences
```

The above command creates an empty database with a table structure similar to Figure 1 and stores it in the file example.sqlite. If a GenDB already exists, then it can be opened using `openGenDB()`.

The next step is to import sequences into the database by reading FASTA files. This is accomplished by function `addSequences()`. This function automatically extracts the classification information from the FASTA file's description lines. The default is to expect classification in the format used by the Greengenes project, however other meta data readers can be implemented (see manual page for addSequences).

The command below uses a FASTA file provided by the package, hence we use `system.file()` instead of just a string with the file name.

```
R> addSequences(db,
+   system.file("examples/Firmicutes.fasta", package="BioTools"))

Read 100 sequences. Added 100 sequences.
```

After inserting the sequences, various querying and limiting functions can be used to check the data and obtain a subset of the sequences. To get a count of the number of sequences in the database, the function `nSequences()` can be used.

```
R> nSequences(db)

[1] 100
```

The function `getSequences()` returns the sequences as a vector. In the following example we get all sequences in the database and then show the first 50 bases of the first sequence.

```
R> s <- getSequences(db)
R> s

  A DNAStringSet instance of length 100
      width seq                                        names
  [1]  1521 TTTGATCCTGGCTCAGG...CGGCTGGATCACCTCCT 1250
  [2]  1392 ACGGGTGAGTAACGCGT...TTGGGGTGAAGTCGTAA 13651
  [3]  1384 TAGTGGCGGACGGGTGA...TCGAATTTGGGTCAAGT 13652
  [4]  1672 GGCGTGCCTAACACATG...TGTAAACACGACTTCAT 13654
  [5]  1386 ATCTCACCTCTCAATAG...CGAAGGTGGGGTTGGTG 13655
  ...   ... ...
 [96]  1446 ATGCAAGTCGAACGGGG...GGGGCCGATGATTGGGG 13857
 [97]  1511 ATCCTGGCTCAGGACGA...AGTCGTAACAAGGTAGC 13858
 [98]  1544 ATCCTGGCTCAGGACGA...GGTGGATCACCTCCTTC 13860
 [99]  1482 GGACGAACGCTGGCGGC...GCCGATGATTGGGGTGA 13861
[100]  1485 GACGAACGCTGGCGGCG...GAAGTCGTAACAAGGTA 13862

R> length(s)

[1] 100

R> s[[1]]

  1521-letter "DNAString" instance
seq: TTTGATCCTGGCTCAGGACGAACGCTGGCGG...TGTACCGGAAGGTGCGGCTGGATCACCTCCT

R> substr(s[[1]], 1, 50)

  50-letter "DNAString" instance
seq: TTTGATCCTGGCTCAGGACGAACGCTGGCGGCGTGCCTAATGCATGCAAG
```

Sequences in the database can also be filtered using classification information. For example, we can get all sequences of the genus name "Desulfosporomusa" by specifying rank and name.

```
R> s <- getSequences(db, rank="Genus", name="Desulfosporomusa")
R> s
```

```
  A DNAStringSet instance of length 7
    width seq                                          names
[1]  1498 TNGAGAGTTTGATCCTGG...TGGGGCCGATGATCGGGG 13834
[2]  1481 CTGGCGGCGTGCCTAACA...ATTGGGGTGAAGTCGTAA 13836
[3]  1510 GACGAACGCTGGCGGCGT...AGCCGTATCGGAAGGTGC 13839
[4]  1503 ACGCTGGCGGCGTGCCTA...GGTAGCCGTATCGGAAGG 13844
[5]  1503 ACGCTGGCGGCGTGCCTA...GGTAGCCGTATCGGAAGG 13845
[6]  1429 ACGCTGGCGGCGTGCCTA...GAAGCCGGTGGGGTAACC 13846
[7]  1504 ACGCTGGCGGCGTGCCTA...GGTAGCCGTATCGGAAGG 13847
```

To obtain a single sequence, getSequences can be used with rank equal to "id" and supplying the sequence's greengenes ID as the name.

```
R> s <- getSequences(db, rank="id", name="1250")
R> s
```

```
  A DNAStringSet instance of length 1
    width seq                                          names
[1]  1521 TTTGATCCTGGCTCAGGA...GCGGCTGGATCACCTCCT 1250
```

The database also stores a classification hierarchy. We can obtain the classification hierarchy used in the database with `getTaxonomyNames()`.

```
R> getTaxonomyNames(db)
```

```
 [1] "Kingdom"  "Phylum"   "Class"    "Order"    "Family"   "Genus"
 [7] "Species"  "Otu"      "Org_name" "Id"
```

To obtain all unique names stored in the database for a given rank we can use `getRank()`.

```
R> getRank(db, rank="Order")
```

```
[1] "Thermoanaerobacterales" "Clostridiales"
```

The 100 sequences in our example data base contain organisms from different orders. We can obtain the rank name for each sequence individually by using `all=TRUE` or count how many sequences we have for each genus using `count=TRUE`.

```
R> getRank(db, rank="Genus", all=TRUE)
```

```
 [1] Coprothermobacter     Desulfotomaculum
 [3] Desulfotomaculum      Desulfotomaculum
 [5] Desulfotomaculum      Desulfotomaculum
 [7] Desulfotomaculum      Desulfotomaculum
 [9] Desulfotomaculum      Pelotomaculum
[11] Desulfotomaculum      Desulfotomaculum
[13] Pelotomaculum         Desulfotomaculum
[15] Desulfotomaculum      Desulfotomaculum
[17] Desulfotomaculum      Pelotomaculum
[19] Desulfotomaculum      Desulfotomaculum
[21] Desulfotomaculum      Desulfotomaculum
[23] Desulfotomaculum      Desulfotomaculum
[25] Pelotomaculum         Syntrophomonas
[27] Syntrophomonas        Syntrophomonas
[29] Syntrophomonas        Syntrophomonas
[31] unknown               Syntrophomonas
[33] Moorella              Moorella
[35] Moorella              Moorella
[37] Thermacetogenium      Thermaerobacter
[39] Carboxydothermus      Carboxydothermus
[41] Thermoanaerobacterium Thermoanaerobacterium
[43] Thermoanaerobacterium Thermoanaerobacterium
[45] Thermoanaerobacterium Thermoanaerobacterium
[47] Thermoanaerobacterium Thermoanaerobacterium
[49] Thermoanaerobacter    Thermoanaerobacter
[51] Thermoanaerobacter    Thermoanaerobacter
[53] Thermoanaerobacter    Thermoanaerobacter
[55] Thermoanaerobacter    Thermoanaerobacter
[57] Thermoanaerobacter    Thermoanaerobacter
[59] Selenomonas           Selenomonas
[61] Selenomonas           Selenomonas
[63] Selenomonas           Mitsuokella
[65] Selenomonas           Selenomonas
[67] Selenomonas           unknown
[69] Selenomonas           Veillonella
[71] Veillonella           Veillonella
[73] Veillonella           Veillonella
[75] Dialister             Dialister
[77] Dialister             Desulfosporomusa
[79] Desulfosporomusa      unknown
[81] unknown               Desulfosporomusa
[83] Thermosinus           Thermosinus
[85] unknown               Desulfosporomusa
[87] Desulfosporomusa      Desulfosporomusa
[89] Desulfosporomusa      unknown
[91] unknown               Acidaminococcus
[93] Acidaminococcus       unknown
```

```
 [95] unknown               unknown
 [97] Phascolarctobacterium Phascolarctobacterium
 [99] unknown               unknown
19 Levels: Acidaminococcus Carboxydothermus ... Veillonella
```

```
R> getRank(db, rank="Genus", count=TRUE)
```

```
     Desulfotomaculum               unknown    Thermoanaerobacter
                   20                    12                    10
         Selenomonas Thermoanaerobacterium      Desulfosporomusa
                    9                     8                     7
       Syntrophomonas           Veillonella             Moorella
                    6                     5                     4
        Pelotomaculum             Dialister       Acidaminococcus
                    4                     3                     2
     Carboxydothermus Phascolarctobacterium           Thermosinus
                    2                     2                     2
    Coprothermobacter           Mitsuokella      Thermacetogenium
                    1                     1                     1
      Thermaerobacter
                    1
```

This information can be easily turned into a barplot showing the abundance of different orders in the data database (see Figure 3).

```
R> oldpar <- par(mar=c(12,5,5,5)) ### make space for labels
R> barplot(sort(
+     getRank(db, rank="Genus", count=TRUE, removeUnknown=TRUE),
+     decreasing=TRUE), las=2)
R> par(oldpar)
```

Filtering also works for `getRank()`. For example, we can find the genera within the order "Thermoanaerobacterales".

```
R> getRank(db, rank="Genus",
+     whereRank="Order", whereName="Thermo")
```

```
[1] "Coprothermobacter"  "Moorella"           "Thermacetogenium"
[4] "Carboxydothermus"   "Thermoanaerobacter"
```

Note that partial matching is performed from "Thermo" to "Thermoanaerobacterales." Partial matching is available for ranks and names in most operations in **BioTools**.

We can also get the complete classification hierarchy for different ranks down to individual sequences. In the following we get the classification hierarchy for genus Thermaerobacter, then all orders matching Therm and then for a list of names.

```
R> getHierarchy(db, rank="Genus", name="Thermaerobacter")
```

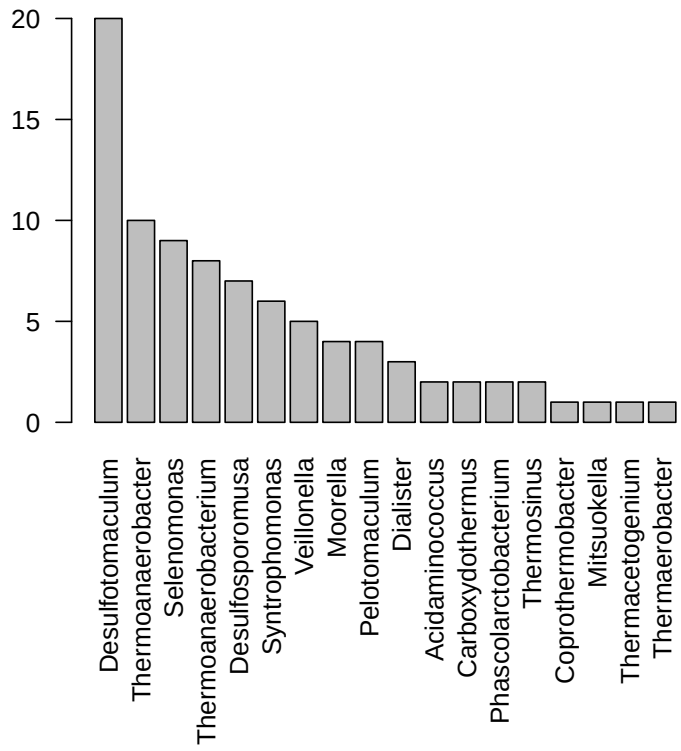

Figure 2: Abundance of different orders in the database.

```
          Kingdom             Phylum              Class
        "Bacteria"       "Firmicutes"       "Clostridia"
            Order              Family              Genus
  "Clostridiales" "Sulfobacillaceae"  "Thermaerobacter"
          Species                Otu           Org_name
               NA                 NA                 NA
               Id
               NA
```

```
R> getHierarchy(db, rank="Genus", name="Therm")
```

```
     Kingdom    Phylum       Class        Order
[1,] "Bacteria" "Firmicutes" "Clostridia" "Thermoanaerobacterales"
[2,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
[3,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
[4,] "Bacteria" "Firmicutes" "Clostridia" "Thermoanaerobacterales"
[5,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
     Family
[1,] "Thermoanaerobacteraceae"
[2,] "Sulfobacillaceae"
[3,] "Thermoanaerobacterales Family III. Incertae Sedis"
[4,] "Thermoanaerobacteraceae"
[5,] "Veillonellaceae"
     Genus                    Species Otu Org_name Id
[1,] "Thermacetogenium"       NA      NA  NA       NA
[2,] "Thermaerobacter"        NA      NA  NA       NA
[3,] "Thermoanaerobacterium" NA      NA  NA       NA
[4,] "Thermoanaerobacter"     NA      NA  NA       NA
[5,] "Thermosinus"            NA      NA  NA       NA
```

```
R> getHierarchy(db, rank="Genus", name=c("Acid", "Thermo"))
```

```
     Kingdom    Phylum       Class        Order
[1,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
[2,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
[3,] "Bacteria" "Firmicutes" "Clostridia" "Thermoanaerobacterales"
[4,] "Bacteria" "Firmicutes" "Clostridia" "Clostridiales"
     Family
[1,] "Veillonellaceae"
[2,] "Thermoanaerobacterales Family III. Incertae Sedis"
[3,] "Thermoanaerobacteraceae"
[4,] "Veillonellaceae"
     Genus                    Species Otu Org_name Id
[1,] "Acidaminococcus"        NA      NA  NA       NA
[2,] "Thermoanaerobacterium" NA      NA  NA       NA
[3,] "Thermoanaerobacter"     NA      NA  NA       NA
[4,] "Thermosinus"            NA      NA  NA       NA
```

To get individual sequences we can use again the unique sequence id.

```
R> getHierarchy(db, rank="id", name="1250")
```

```
                                                Kingdom
                                              "Bacteria"
                                                 Phylum
                                             "Firmicutes"
                                                  Class
                                             "Clostridia"
                                                  Order
                                  "Thermoanaerobacterales"
                                                 Family
                                    "Thermodesulfobiaceae"
                                                  Genus
                                       "Coprothermobacter"
                                                Species
                                              "unknown"
                                                    Otu
                                             "otu_2281"
                                               Org_name
"X69335.1Coprothermobacterproteolyticusstr.ATCC35245"
                                                   Id
                                               "1250"
```

Finally, we can close a GenDB after we are done working with it. The database can later be reopened using `openGenDB()`.

```
R> closeGenDB(db)
```

To permanently remove the database we need to delete the file (for SQLite databases) or remove the database using the administrative tool for the database management system.

```
R> unlink("example.sqlite")
```

FIXME: Is there a purge function in DBI to do this?

3. Multiple Sequence Alignment

3.1. clustalw

```
R> dna <- readDNAStringSet(system.file("examples/DNA_example.fasta",
+         package="BioTools"))
R> dna <- narrow(dna, start=1, end=60)
R> al <- clustal(dna)
R> al
```

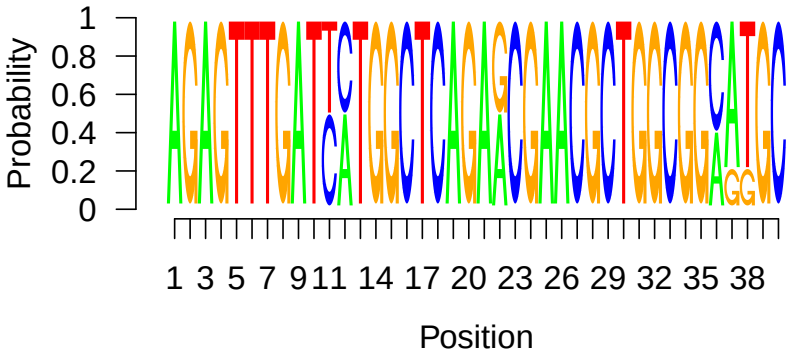

Figure 3: Sequence logo of alignment.

```
4403    ---  ------------------------------GGAATGCTNAACACATGCAAGTCGCACGG--
4404    ---  ------------------------GCTGGCGGAATGCTTAACACATGCAAGTCGCACGGGG
4399    ---  ------------------------GCTGGCGGCAGGCCTAACACATGCAAGTCGAACGGGG
1675    AGA  TTTGATCCTGGCTCAGAACGAACGCTGGCGGCGTGCCTAACACATGCAAGTCGAAC----
4411    AGA  TTTGATTATGGCTCAGAGCGAACGCTGGCGGCATGCTTAACACATGCAAGTCGCAC----
consensus                           gctggcGGcatGCttAACACATGCAAGTCGcACgg
           1........10........20........30........40........50.........

4403    ---  GCAGC--AATGTCA-GTGGCGGACGGGTGAGTAA
4404    ---  GTTTC--GGCCTTA-GTGGCGGACGG--------
4399    ---  ACCTTCGGGTCTTACGTGGCGCA-----------
1675    AGA  ----------------------------------
4411    AGA  ----------------------------------
consensus    g      aa  t a gtggcg a
           61.......70........80........90...
```

Figure 4: Representaion of a DNA multiple alignment using boxshade.

```
DNAMultipleAlignment with 5 rows and 98 columns
     aln                                                 names
[1] ----------------------...-GTGGCGGACGGGTGAGTAA 4403
[2] ----------------------...-GTGGCGGACGG-------- 4404
[3] ----------------------...CGTGGCGCA----------- 4399
[4] AGAGTTTGATCCTGGCTCAGA...-------------------- 1675
[5] AGAGTTTGATTATGGCTCAGA...-------------------- 4411

R> ### inspect alignment
R> detail(al)

R> plot(al, 1, 40)


R> boxshade(al, file="alignment.pdf")


R> rna <- readRNAStringSet(system.file("examples/RNA_example.fasta",
+         package="BioTools"))
R> rna
```

```
  A RNAStringSet instance of length 5
    width seq                                          names
[1]  1481 AGAGUUUGAUCCUGGCUC...AGUCGUAACAAGGUAACC 1675 AB015560.1 d...
[2]  1404 GCUGGCGGCAGGCCUAAC...UAAGGUCAGCGACUGGGG 4399 D14432.1 Rho...
[3]  1426 GGAAUGCUNAACACAUGC...GGUAGCCGUAGGGGAACC 4403 X72908.1 Ros...
[4]  1362 GCUGGCGGAAUGCUUAAC...UAGGUGUCUAGGCUAACC 4404 AF173825.1 A...
[5]  1458 AGAGUUUGAUUAUGGCUC...UCGUAACAAGGUAACCGU 4411 Y07647.2 Dre...

R> al <- clustal(rna)
R> al

RNAMultipleAlignment with 5 rows and 1500 columns
     aln                                           names
[1] ----------...AAGGUAGCCGUAGGGGAACC 4403
[2] ----------...---------- 4404
[3] AGAGUUUGAUUAUGGCUCAGA...AAGGUAACCGU--------- 4411
[4] ----------...---------- 4399
[5] AGAGUUUGAUCCUGGCUCAGA...AAGGUAACC----------- 1675

R> aa <- readAAStringSet(system.file("examples/Protein_example.fasta",
+         package="BioTools"))
R> aa

  A AAStringSet instance of length 5
    width seq                                          names
[1]   170 MKKSWRRIWIFGLLFSIW...DVYYLEAPFFQGRKCGGT gi|340754543|ref|...
[2]   233 MYIIWKLLFFKGENVVEH...KEEEVISVVDDILKKRRE gi|340754544|ref|...
[3]   326 MKRSLSGIQPSGILHLGN...KKVQEAKEIVGLLGNIYR gi|340754545|ref|...
[4]   317 MKYYSGVDLGGTNTKIGL...VLGNEAGILGAAALFMLS gi|340754546|ref|...
[5]   337 MKKMGIILGALVLAAGLV...IVLVPSIGIDKENVAEYK gi|340754547|ref|...

R> al <- clustal(aa)
R> al

AAMultipleAlignment with 5 rows and 358 columns
     aln                                           names
[1] ---MKKSWRRIWIFGLLFSIW...---------- gi|340754543|ref|...
[2] ---MYIIWKLLFFKGENVVEH...---------- gi|340754544|ref|...
[3] MKKMGIILGALVLAAGLVGCG...DKENVAEYK----------- gi|340754547|ref|...
[4] ---MKRSLSGIQPSGILHLGN...ASKKVQEAKEIVGLLGNIYR gi|340754545|ref|...
[5] ----MKYYSGVDLGGTNTKIG...---------- gi|340754546|ref|...
```

3.2. kalign

```
R> dna <- readDNAStringSet(system.file("examples/DNA_example.fasta",
+         package="BioTools"))
R> dna
```

```
  A DNAStringSet instance of length 5
    width seq                                          names
[1]  1481 AGAGTTTGATCCTGGCTC...AGTCGTAACAAGGTAACC 1675 AB015560.1 d...
[2]  1404 GCTGGCGGCAGGCCTAAC...TAAGGTCAGCGACTGGGG 4399 D14432.1 Rho...
[3]  1426 GGAATGCTNAACACATGC...GGTAGCCGTAGGGGAACC 4403 X72908.1 Ros...
[4]  1362 GCTGGCGGAATGCTTAAC...TAGGTGTCTAGGCTAACC 4404 AF173825.1 A...
[5]  1458 AGAGTTTGATTATGGCTC...TCGTAACAAGGTAACCGT 4411 Y07647.2 Dre...

R> ### align the sequences
R> al <- kalign(dna)
R> al

DNAMultipleAlignment with 5 rows and 1502 columns
     aln                                           names
[1] AGAGTTTGATCCTGGCTCAGA...--------CAAGGTAAC--C 1675 AB015560.1 d...
[2] G--------------------...--------TGGG-------G 4399 D14432.1 Rho...
[3] G--------------------...GGTAGCCGTAGGGGAAC--C 4403 X72908.1 Ros...
[4] G--------------------...--------TAGGCTAAC--C 4404 AF173825.1 A...
[5] AGAGTTTGATTATGGCTCAGA...--------CAAGGTAACCGT 4411 Y07647.2 Dre...
```

4. Classification with RDP

4.1. Using the defaul RDP classifier

Use the default classifier

```
R> seq <- readRNAStringSet(system.file("examples/RNA_example.fasta",
+         package="BioTools"))
R> ## shorten names
R> names(seq) <-  sapply(strsplit(names(seq), " "), "[", 1)
R> seq

  A RNAStringSet instance of length 5
    width seq                                          names
[1]  1481 AGAGUUUGAUCCUGGCUC...AGUCGUAACAAGGUAACC 1675
[2]  1404 GCUGGCGGCAGGCCUAAC...UAAGGUCAGCGACUGGGG 4399
[3]  1426 GGAAUGCUNAACACAUGC...GGUAGCCGUAGGGGAACC 4403
[4]  1362 GCUGGCGGAAUGCUUAAC...UAGGUGUCUAGGCUAACC 4404
[5]  1458 AGAGUUUGAUUAUGGCUC...UCGUAACAAGGUAACCGU 4411

R> ## use rdp for classification
R> predict(RDP(), seq)

     norank   domain         phylum               class
1675   Root Bacteria Proteobacteria Deltaproteobacteria
```

```
4399   Root Bacteria Proteobacteria Alphaproteobacteria
4403   Root Bacteria Proteobacteria Alphaproteobacteria
4404   Root Bacteria Proteobacteria Alphaproteobacteria
4411   Root Bacteria Proteobacteria Alphaproteobacteria
               order            family       genus
1675            <NA>              <NA>        <NA>
4399 Rhodospirillales Rhodospirillaceae Rhodovibrio
4403 Rhodospirillales  Acetobacteraceae Roseococcus
4404 Rhodospirillales  Acetobacteraceae Roseococcus
4411 Rhodospirillales  Acetobacteraceae        <NA>
```

4.2. Training a custom RDP classifier

Train a custom RDP classifier on new data

```
R> trainingSequences <- readDNAStringSet(
+     system.file("examples/trainingSequences.fasta", package="BioTools"))
R> customRDP <- trainRDP(trainingSequences)

Called from: trainRDP(trainingSequences)

R> customRDP

RDPClassifier
Location: /home/hahsler/baR/QuasiAlign/pkg/BioTools/Work/vignette/classifier

R> testSequences <- readDNAStringSet(
+     system.file("examples/testSequences.fasta", package="BioTools"))
R> predict(customRDP, testSequences)

      rootrank  Kingdom     Phylum      Class         Order
13811     Root Bacteria Firmicutes Clostridia Clostridiales
13813     Root Bacteria Firmicutes Clostridia Clostridiales
13678     Root Bacteria Firmicutes Clostridia Clostridiales
13755     Root Bacteria Firmicutes Clostridia Clostridiales
13661     Root Bacteria Firmicutes Clostridia Clostridiales
                                                    Family
13811                                       Veillonellaceae
13813                                       Veillonellaceae
13678                                        Peptococcaceae
13755 Thermoanaerobacterales Family III. Incertae Sedis
13661                                        Peptococcaceae
                     Genus
13811           Selenomonas
13813           Selenomonas
13678      Desulfotomaculum
13755 Thermoanaerobacterium
13661      Desulfotomaculum
```

```
R> ## clean up
R> removeRDP(customRDP)
```

5. Sequence Retrieval with BLAST

```
R> seq <- readRNAStringSet(system.file("examples/RNA_example.fasta",
+         package="BioTools"))
R> ## shorten names
R> names(seq) <-  sapply(strsplit(names(seq), " "), "[", 1)
R> seq

  A RNAStringSet instance of length 5
    width seq                                          names
[1]  1481 AGAGUUUGAUCCUGGCUC...AGUCGUAACAAGGUAACC 1675
[2]  1404 GCUGGCGGCAGGCCUAAC...UAAGGUCAGCGACUGGGG 4399
[3]  1426 GGAAUGCUNAACACAUGC...GGUAGCCGUAGGGGAACC 4403
[4]  1362 GCUGGCGGAAUGCUUAAC...UAGGUGUCUAGGCUAACC 4404
[5]  1458 AGAGUUUGAUUAUGGCUC...UCGUAACAAGGUAACCGU 4411

R> ## load a BLAST database (replace db with the location + name of the BLAST DB)
R> blast <- BLAST(db="~/tmp/blast/16SMicrobial")
R> blast

BLAST Database
Location: /home/hahsler/tmp/blast/16SMicrobial

R> print(blast, info=TRUE)

BLAST Database
Location: /home/hahsler/tmp/blast/16SMicrobial
Database:  16S Microbial Sequences
        8,412 sequences; 12,354,954 total bases

Date: Mar 26, 2013  12:51 AM        Longest sequence: 1,768 bases

Volumes:
        /home/hahsler/tmp/blast/16SMicrobial

R> ## query a sequence using BLAST
R> cl <- predict(blast, seq[1,])
R> cl[1:5,]

  QueryID                     SubjectID Perc.Ident Alignment.Length
1    1675 gi|444304125|ref|NR_074549.1|      86.11             1260
```

```
2    1675 gi|444304125|ref|NR_074549.1|      94.20              69
3    1675 gi|343201138|ref|NR_041853.1|      82.57            1555
4    1675 gi|444439604|ref|NR_074919.1|      82.50            1543
5    1675 gi|265678428|ref|NR_028730.1|      82.82            1519
  Mismatches Gap.Openings Q.start Q.end S.start S.end     E Bits
1        136           36     235  1478     247  1483 0e+00 1321
2          4            0       1    69       1    69 2e-22  106
3        162           83       3  1481       1  1522 0e+00 1269
4        177           67       1  1478       1  1515 0e+00 1267
5        156           81      31  1475       1  1488 0e+00 1267
```

6. Conclusion

Acknowledgments

This research is supported by research grant no. R21HG005912 from the National Human Genome Research Institute (NHGRI / NIH).

References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). "Basic local alignment search tool." *Journal of Molecular Biology*, **215**(3), 403–410. ISSN 0022-2836.

Edgar R (2004a). "MUSCLE: a multiple sequence alignment method with reduced time and space complexity." *BMC Bioinformatics*, **5**(1), 113+. ISSN 1471-2105.

Edgar RC (2004b). "Muscle: multiple sequence alignment with high accuracy and high throughput." *Nucleic Acids Research*, **32**, 1792–1797.

Katoh K, Misawa K, Kuma K, Miyata T (2002). "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform." *Nucleic Acids Research*, **30**(14), 3059–3066.

Larkin M, Blackshields G, Brown N, Chenna R, McGettigan P, McWilliam H, Valentin F, Wallace I, Wilm A, Lopez R, Thompson J, Gibson T, Higgins D (2007). "Clustal W and Clustal X version 2.0." *Bioinformatics*, **23**, 2947–2948. ISSN 1367-4803.

Lassmann T, Sonnhammer EL (2006). "Kalign, Kalignvu and Mumsa: web servers for multiple sequence alignment." *Nucleic Acids Research*, **34**. ISSN 1362-4962.

Notredame C, Higgins DG, Heringa J (2000). "T-Coffee: A novel method for fast and accurate multiple sequence alignment." *Journal of Molecular Biology*, **302**(1), 205–217. ISSN 0022-2836.

Smith TF, Waterman MS (1981). "Identification of common molecular subsequences." *Journal of Molecular Biology*, **147**(1), 195–197. ISSN 0022-2836.

Affiliation:

Michael Hahsler
Engineering Management, Information, and Systems
Lyle School of Engineering
Southern Methodist University
P.O. Box 750123
Dallas, TX 75275-0123
E-mail: mhahsler@lyle.smu.edu
URL: http://lyle.smu.edu/~mhahsler

Anurag Nagar
Computer Science and Engineering
Lyle School of Engineering
Southern Methodist University
P.O. Box 750122
Dallas, TX 75275-0122
E-mail: anagar@smu.edu