Getting Started DECIPHERing

Erik S. Wright
University of Wisconsin
Madison, WI

March 22, 2013

Contents

1 About DECIPHER

*D*atabase *E*nabled *C*ode for *I*deal *P*robe *H*ybridization *E*mploying *R* (DECIPHER) is a software toolset that can be used for deciphering and managing DNA sequences efficiently using the R statistical programming language. The project originally sprang to life as a program for developing hybridization probes for a variety of applications using 16S rRNA sequences. Although the program's functionality has expanded since its conception, it still maintains the name DECIPHER to this day. DECIPHER is available under the terms of the GNU Public License version 3 (`http://www.gnu.org/copyleft/gpl.html`).

2 Design Philosophy

2.1 Curators Protect the Originals

One of the core principles of DECIPHER is the idea of the non-destructive workflow. This revolves around the concept that the original sequence information should never be altered: sequences are exported looking identical to how they were when they were first imported. Essentially, the sequence information in the database is thought of as a backup of the original sequence file and no function is able to directly alter the sequence data. All of the workflows simply *add* information to the database, which can be used to maintain, analyze, and decipher the sequences. When it comes time to export all or part of the sequences they are preserved in their original state without alteration.

2.2 Don't Reinvent the Wheel

DECIPHER makes use of the Biostrings package that is a core part of the Bioconductor suite (`http://www.bioconductor.org/`). This package contains numerous functions for common operations such as searching, aligning, and reverse complementing sequences. Furthermore, DECIPHER makes use of the Biostrings interface for handling DNA sequence data so that sequences are stored in a `DNAStringSet`. These objects are compatible with many useful packages in the Bioconductor suite.

A wide variety of user objectives necessitates that DECIPHER be extensible to customized projects. R provides a simple way to place the power of thousands of packages at your fingertips. Likewise, R enables direct access to the speed and efficiency of the programming language C while maintaining the utility of a scripting language. Therefore, minimal coding skill is required to solve complex new problems. Best of all, the R statistical programming language is open source, and maintains a thriving user community so that direct collaboration with other R users is available on several Internet forums `https://stat.ethz.ch/mailman/listinfo`.

2.3 That Which is the Most Difficult, Make Fastest

A core objective of DECIPHER is to make massive tasks feasible in minimal time. To this end, many of the most time consuming functions are parallelized to make use of multiple processors. For example, the function `DistanceMatrix` gets almost a 1x speed boost for each processor core. A modern processor with 8 cores can see a factor of close to eight times speed improvement. Similar speedups can be achieved when clustering the resulting distance matrix using `IdClusters`. This is all made possible through the integration of OpenMp, which is currently supported by default on most major platforms expect Windows (see installation 4).

Other time consuming tasks are handled efficiently. The function `FindChimeras` can uncover sequence chimeras by searching through a reference database of over a million sequences for thousands of 30-mer fragments in a number of minutes. This incredible feat is accomplished by using the *PDict* class provided by Biostrings. Similarly, the `SearchDB` function can obtain the one-in-a-million sequences that match a targeted query in a matter of seconds. Such high-speed functions enable the user to find solutions to problems that previously would have been extremely difficult or nearly impossible to solve using antiquated methods.

2.4 Stay Organized

It is no longer necessary to store related data in several different files. DECIPHER is enabled by RSQLite, which is an R interface to *SQLite* databases `http://www.sqlite.org/`. DECIPHER creates an organized collection of sequences and their associated information known as a sequence database. *SQLite* databases are flat files, meaning they can be handled just like any other file. There is no setup required since *SQLite* does not require a server, unlike other client database engines. These attributes of *SQLite* databases make sharing, backing-up, and storing sequence databases relatively straightforward.

Separate projects can be stored in distinct tables in the same sequence database. Each new table is structured to include every sequence's description, identifier, and a unique key called the *row_name* all in one place. The sequences are referenced by their *row_names* or *identifier* throughout most functions in the package. New information created using DECIPHER functions is added as additional database columns to its respective sequences' *row_names*. To prevent the database from seeming like a black box there is a function named `BrowseDB` that facilitates viewing of the database contents in a web browser. A similar function is available to view sequences called `BrowseSequences`.

The amount of DNA sequence information available is currently increasing at a phenomenal rate. DECIPHER stores individual sequences using *gzip* compression so that the database file takes up much less drive space than a standard text file of sequences. The compressed sequences are stored in a hidden table that is linked to the information table. Storing the sequences in a separate table greatly improves access speed when there is a large amount of sequence information. Separating projects into distinct tables further increases query speed over that of storing every project in a single table.

3 Functionality

The functions of DECIPHER can be grouped into several categories based on intended use:

1. Primary functions for interacting with a sequence database:
   (a) `Add2DB`
   (b) `DB2FASTA`
   (c) `SearchDB`
   (d) `Seqs2DB`
2. Secondary functions for typical database tasks:
   (a) `IdConsensus`
   (b) `IdentifyByRank`
   (c) `IdLengths`
3. Functions for phylogenetics with a set of DNA sequences:
   (a) `ConsensusSequence`
   (b) `DistanceMatrix`
   (c) `IdClusters`
4. Functions for visualization with a web browser:
   (a) `BrowseDB`
   (b) `BrowseSequences`
5. Functions related to chimeras:
   (a) `CreateChimeras`
   (b) `FindChimeras`
   (c) `FormGroups`
6. Functions related to DNA microarrays:
   (a) `CalculateEfficiencyArray`
7. Functions related to primers for polymerase chain reaction (PCR):
   (a) `CalculateEfficiencyPCR`
   (b) `DesignPrimers`
   (c) `TileSeqs`

4 Installation

4.1 Typical Installation (recommended)

1. Install R (version >= 2.13.0) from `http://www.r-project.org/`.
2. Install DECIPHER in R by entering:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("DECIPHER")
```

4.2 Manual Installation

4.2.1 All platforms

1. Install R (version >= 2.13.0) from `http://www.r-project.org/`.
2. Install Biostrings in R by entering:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("Biostrings")
```

3. Install RSQLite in R by entering:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("RSQLite")
```

4. Download DECIPHER from `http://DECIPHER.cee.wisc.edu`.

4.2.2 Mac OS X

```
> install.packages("<<path to Mac OS X DECIPHER.tgz>>", repos=NULL)
```

4.2.3 Linux

In a shell enter:

```
R CMD build --no-vignettes "<<path to DECIPHER source>>"
R CMD INSTALL "<<path to newly built DECIPHER.tar.gz>>"
```

4.2.4 Windows

Two options are available: the first is simpler but does not have support for making use of multiple processors.

1. First Option (simplest but no parallelization):

```
> install.packages("<<path to Windows DECIPHER.zip>>", repos=NULL)
```

2. Second Option (more difficult but enables parallelization):
   (a) Download pcthreadsGC2.dll from `http://sourceware.org/pthreads-win32/`.
   (b) Place pcthreadsGC2.dll in a place where R will find it such as the R/bin/ folder inside Program Files.
   (c) Install Rtools from `http://www.murdoch-sutherland.com/Rtools/`. Be sure to check the box that says edit PATH during installation.

(d) Open the DECIPHER source folder. Add a text file named "Makevars.win" into the "src" folder with the following two lines:

```
PKG_CFLAGS=-fopenmp
PKG_LIBS=-mthreads -lgomp -lpthreadGC2
```

(e) Open a MS-DOS command prompt by clicking Start -> All Programs -> Accessories >- Command Prompt.

(f) In the command prompt, enter:

```
R CMD build --no-vignettes "<<path to DECIPHER source>>"
R CMD INSTALL "<<path to newly built DECIPHER.zip>>"
```

5 Example Workflow

To get started we need to load the DECIPHER package, which automatically loads several other required packages:

```
> library(DECIPHER)
```

Help for any function can be accessed through a command such as:

```
> ? DECIPHER
```

To begin we can import a GenBank file of sequences into a sequence database. We need to provide an arbitrary sequence *identifier* to `Seqs2DB`, which we will call "Bacteria". The *identifier* is used by many DECIPHER functions to reference a specific set of sequences in the database:

```
> # access a sequence file included in the package:
> gen <- system.file("extdata", "Bacteria_175seqs.gen", package="DECIPHER")
> # connect to a database:
> dbConn <- dbConnect(SQLite(), ":memory:")
> # import the sequences into the sequence database
> Seqs2DB(gen, "GenBank", dbConn, "Bacteria")
```

```
Reading GenBank file from line 0 to 99999

175 total sequences in database.
Time difference of 0.36 secs

[1] 175
```

Now we can view the table of information we just added to the database in a web browser:

```
> BrowseDB(dbConn)
```

```
[1] TRUE
```

Suppose we wanted to count the number of bases in each sequence and add that information to the database:

```
> l <- IdLengths(dbConn)
```

```
Lengths counted for 175 sequences.
```

```
> head(l)
```

```
  bases nonbases width
1  1222       13  1596
2  1346        5  1596
3  1323        3  1596
4  1337        8  1596
5  1318       25  1596
6  1307        7  1596
```

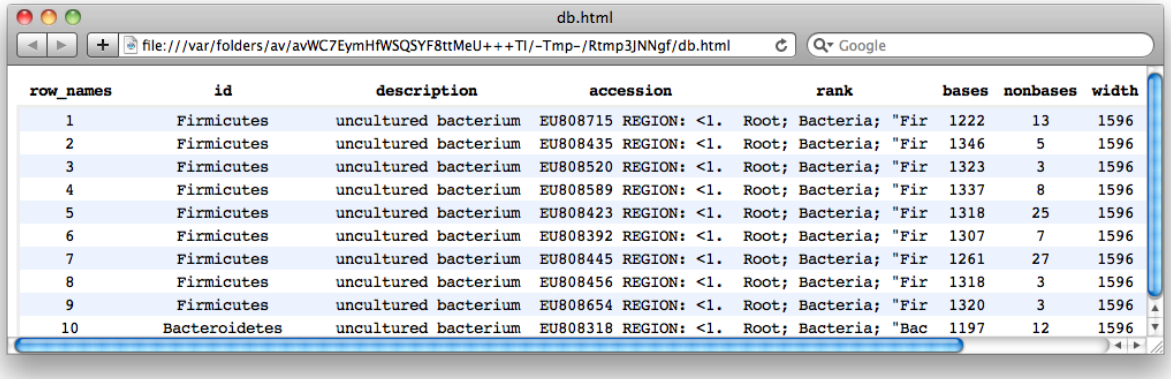
```
> Add2DB(l, dbConn)

Expression:  alter table DNA add column bases INTEGER
Expression:  update or replace DNA set bases = :bases where row_names = :row_names

Expression:  alter table DNA add column nonbases INTEGER
Expression:  update or replace DNA set nonbases = :nonbases where row_names = :row_names

Expression:  alter table DNA add column width INTEGER
Expression:  update or replace DNA set width = :width where row_names = :row_names

Added to table DNA:  "bases" and "nonbases" and "width".
Time difference of 0.02 secs

[1] TRUE

> BrowseDB(dbConn, maxChars=20)

[1] TRUE
```

Next let's identify our sequences by phylum and update this information in the database:

```
> r <- IdentifyByRank(dbConn, add2tbl=TRUE)

Updating column: "id"...
Formed 7 distinct groups.
Added to table DNA: "id".
Time difference of 0.03 secs

> BrowseDB(dbConn, maxChars=20)

[1] TRUE
```

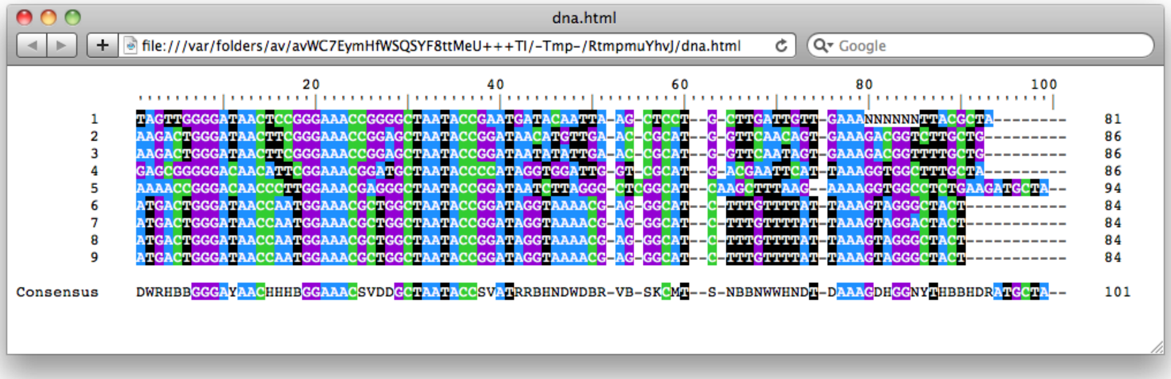
We can now look at only those sequences that belong to the phylum *Firmicutes*:

```
> dna <- SearchDB(dbConn, id="Firmicutes")

Search Expression:
select row_names, sequence from _DNA where row_names in (select row_names from DNA where id like "Firmicu

DNA Stringset of length: 9
Time difference of 0.01 secs

> BrowseSequences(subseq(dna, 140, 240), colorBases=TRUE)

[1] TRUE
```



row_names	id	description	accession	rank	bases	nonbases	width
1	Firmicutes	uncultured bacterium	EU808715 REGION: <1.	Root; Bacteria; "Fir	1222	13	1596
2	Firmicutes	uncultured bacterium	EU808435 REGION: <1.	Root; Bacteria; "Fir	1346	5	1596
3	Firmicutes	uncultured bacterium	EU808520 REGION: <1.	Root; Bacteria; "Fir	1323	3	1596
4	Firmicutes	uncultured bacterium	EU808589 REGION: <1.	Root; Bacteria; "Fir	1337	8	1596
5	Firmicutes	uncultured bacterium	EU808423 REGION: <1.	Root; Bacteria; "Fir	1318	25	1596
6	Firmicutes	uncultured bacterium	EU808392 REGION: <1.	Root; Bacteria; "Fir	1307	7	1596
7	Firmicutes	uncultured bacterium	EU808445 REGION: <1.	Root; Bacteria; "Fir	1261	27	1596
8	Firmicutes	uncultured bacterium	EU808456 REGION: <1.	Root; Bacteria; "Fir	1318	3	1596
9	Firmicutes	uncultured bacterium	EU808654 REGION: <1.	Root; Bacteria; "Fir	1320	3	1596
10	Bacteroidetes	uncultured bacterium	EU808318 REGION: <1.	Root; Bacteria; "Bac	1197	12	1596

Figure 1: Database table shown in web browser

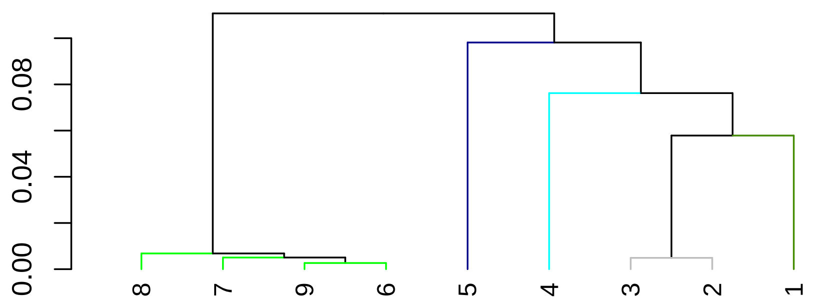


```
                   20                  40                  60                  80                  100
1          TAGTTGGGGATAACTCCGGGAAACCGGGGCTAATACCGAATGATACAATTA-AG-CTCCT--G-CTTGATTGTT-GAAANNNNNNTTACGCTA---------   81
2          AAGACTGGGATAACTTCGGGAAACCGGAGCTAATACCGGATAACATGTTGA-AC-CGCAT--G-GTTCAACAGT-GAAAGACGGTCTTGCTG---------   86
3          AAGACTGGGATAACTTCGGGAAACCGGAGCTAATACCGGATAATATATTGA-AC-CGCAT--G-GTTCAATAGT-GAAAGACGGTTTTGCTG---------   86
4          GAGCGGGGGACAACATTCGGAAACGGATGCTAATACCCCATAGGTGGATTG-GT-CGCAT--G-ACGAATTCAT-TAAAGGTGGCTTTGCTA---------   86
5          AAAACCGGGACAACCCTTGGAAACGAGGGCTAATACCGGATAATCTTAGGG-CTCGGCAT--CAAGCTTTAAG--AAAAGGTGGCCTCTGAAGATGCTA--   94
6          ATGACTGGGATAACCAATGGAAACGCTGGCTAATACCGGATAGGTAAAACG-AG-GGCAT--C-TTTGTTTTAT-TAAAGTAGGGCTACT-----------   84
7          ATGACTGGGATAACCAATGGAAACGCTGGCTAATACCGGATAGGTAAAACG-AG-GGCAT--C-TTTGTTTTAT-TAAAGTAGGACTACT-----------   84
8          ATGACTGGGATAACCAATGGAAACGCTGGCTAATACCGGATAGGTAAAACG-AG-GGCAT--C-TTTGTTTTAT-TAAAGTAGGGCTACT-----------   84
9          ATGACTGGGATAACCAATGGAAACGCTGGCTAATACCGGATAGGTAAAACG-AG-GGCAT--C-TTTGTTTTAT-TAAAGTAGGGCTACT-----------   84

Consensus  DWRHBBGGGAYAACHHHBGGAAACSVDDGCTAATACCSVATRRBHNDWDBR-VB-SKCMT--S-NBBNWWHNDT-DAAAGDHGGNYTHBBHDRATGCTA--  101
```

Figure 2: Sequences shown in web browser

Let's construct a phylogenetic tree from the *Firmicutes* sequences:

```
> d <- DistanceMatrix(dna, verbose=FALSE)
> c <- IdClusters(d, method="complete", cutoff=.02, showPlot=TRUE, verbose=FALSE)
```



We could then use the command below to save the in-memory database to a file for long term storage. Be sure to change the path names to those on your system by replacing all of the text inside quotes labeled "<<path to ...>>" with the actual path on your system.

```
> sqliteCopyDatabase(dbConn, "<<path to database>>")
```

Finally, we should disconnect from the database connection. Since the sequence database was created in temporary memory, all of the information will be erased:

```
> dbDisconnect(dbConn)
```

```
[1] TRUE
```

6 Session Information

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

- R version 2.15.2 (2012-10-26), `x86_64-apple-darwin9.8.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: BiocGenerics 0.4.0, Biostrings 2.27.3, DBI 0.2-5, DECIPHER 1.5.0, IRanges 1.16.4, RSQLite 0.11.2
- Loaded via a namespace (and not attached): parallel 2.15.2, stats4 2.15.2, tools 2.15.2