MEGAPTERA - Mega-Phylogeny Techniques in R

Christoph Heibl

November 13, 2014

Contents

1 Introduction

This package facilitates the automated construction of large DNA sequences datasets from internet-accessible depositories, currently the Nucleotide database on GenBank (http://www.ncbi.nlm.nih.gov/nuccore). It uses a number of successive steps to query, download, check, sort and assemble DNA sequences. The minimum input of information required by the algorithm is the name of a DNA sequence marker (e.g. rbcL, trnS, cytB, ...) and some taxon names. First, a taxonomic classification for the taxa of interest is retrieved from the Taxonomy database on GenBank. The taxonomy will be used for reference sequences calculation and profile alignment. In the second step, GenBank is searched for all available sequences of a marker for the given set of species. The sequences are downloaded and, if more than one accession is available per species, species alignments are build. In a fourth step, these species alignments are searched for those alignments that contain totally identical 'zero-distance' sequences and a consensus sequence, hereafter termed 'reference sequence', is build from the zero-distance alignments. The algorithm considers the reference sequence as some sort of 'idealized' sequence of the genetic marker, to which, in a fifth step, all sequences are compared and their inclusion into the final alignment is decided upon. Knowing that there is no guarantee that all sequence information stored in the database is correct and lacking *a priori* measures to tell the right from the wrong sequences, the rationale behind this process is that we might be especially confident in the correctness of a sequence, if it is shared by all conspecifics in that database and our confidence might be the greater the more sequences are available for this comparison. Next, all included sequences are aligned and the pairwise-distances of the sequences are calculated. If the maximum genetic distance exceeds a certain threshold, the alignment is iteratively broken into smaller alignment blocks until the condition of the maximum distance threshold is satisfied. The resulting blocks are cleaned and concatenated into the given marker's alignment. Several such alignments can then be concatenated into a supermatrix or analyzed separately with supertree methods.

2 Required software

MEGAPTERA is exclusively written in R and makes extensive use of the existing capabilities of R to deal with phylogenetic data provided by the packages APE, SEQINR, and IPS.

For sequence alignment and for alignment masking, MEGAPTERA uses external software packages, which are usually much faster than pure R code, and we profit in using stable and millionfold-tested software package. Currently, there is only one option for both tasks: sequence alignment is done with MAFFT[1] and the masking of doubtful alignment position with GBLOCKS[2], but further programs might be included in the future. See the respective websites for installation.

Data management relies on POSTGRESQL, a popular open relational database, for which an excellent interface with R is available (packages DBI and RPOSTGRESQL). See the POSTGRESQL website[3] for installation and and a basic introduction to SQL.

3 Preparing the pipeline

MEGAPTERA uses a relational database system to store taxonomies, sequences of different genetic markers and their interrelation. This might first seem unnecessarily complicated for users who are used to handling sequence data files in formats like NEXUS, PHYLIP, or FASTA, but the benefits are clear: A database is much easier to maintain and evolve in a consistent fashion due to

[1]http://align.bmr.kyushu-u.ac.jp/mafft/software/
[2]http://molevol.cmima.csic.es/castresana/Gblocks.html
[3]http://www.postgresql.org

the powerful standard query language (SQL). In addition, database operations are usually much quicker compared to file-based input/output, a difference that is increasingly important when it comes to build larger and larger datasets.

Once the database system is available (3.1), the minimum required information to start the pipeline is a taxonomy of the study group (3.2) and a set of genetic markers (3.3), before the actual pipeline can be run (4).

To show how to use the MEGAPTERA package, we are going to construct a phylogenetic dataset of the mammal order Cetacea (whales and dolphins). With 80–90 extant species, Cetacea present an ideal group for demonstration purposes. And maybe a motivating one, as I hope that many users will share my fascination for this mysterious creatures. Last not least the monotypic genus of humpback whales served as an eponym of this package.

3.1 Set up the database

First we create a database called `cetacea` to hold all sequences together with taxonomic data; the corresponding function is `dbPars()`. In principle, you can do this and all subsequent SQL operations with PostgreSQL's genuine front-end PGADMIN3[4], but I recommend to use MEGAPTERA's functions to assure the integrity of your data. Besides the name of the database (`dbname`), the drivers accepts four parameters: `host`, `port`, `user`, and (maybe) `password`. With a standard installation of postgreSQL you should be fine leaving these parameters at their default values.[5]

```
library(megaptera)
conn <- dbPars(dbname = "Cetacea", password = "oxalis")
```

Note that `dbname` was coerced to lower case, because SQL is not case sensitive and the same applies to all table and attribute (column) names.

```
show(conn)

## PostgreSQL connection parameters:
##       host = localhost
##       port = 5432
##     dbname = cetacea
##       user = postgres
##   password = oxalis
```

3.2 Taxonomic information

Whereas evolutionary biologists are normally interested in the phylogenetic history of a clade, community ecologists need phylogenies of the members of a

[4]http://www.postgresql.org/ftp/pgadmin3/

[5]If you encounter any problems, have a look at `?dbConnect` and the postgreSQL documentation.

community, i.e. only a subset of of a clade's phylogeny. To account for both needs, MEGAPTERA accepts taxonomic information as a species list or as the name of one or more (higher) taxa (e.g. Cetacea, Asteraceae, Russula). Ingroup and outgroup is given separately and we have to specify the kingdom our taxa belong to. This is neccesary to differenciate between homonyms allowed by the taxonomic codices (e.g. *Prunella* ist both a genus of birds and of plants).

```
tax <- taxon(ingroup = "Cetacea",
             outgroup = c("Hippopotamus amphibius",
                          "Sus scrofa",
                          "Bos taurus"),
             kingdom = "Metazoa")

tax

## --- megaptera taxon class ---
## ingroup taxon  : Cetacea
## outgroup taxon : Hippopotamus amphibius, Sus scrofa , ... [ 3 ]
## in kingdom     : Metazoa
## hybrids        : excluded
```

For running the pipeline the taxonomic classification will be stored in a database table called `taxonomy`. You can create this table yourself or you can use `dbUpdateTaxonomy`, which his recommended, because the function takes care of standardizing the names of ranks and will append taxa if `taxonomy` already exists. In the case that you do not have a full taxonomy of your study group at hand, but only a species list without higher taxonomic ranks, you can use the function `ncbiTaxonomy` to retrieve the relevant information from the taxonomy database provided by NCBI. The same function is useful if you are interested in all members of a certain higher taxon. For demostration we will download the classification of the order Cetacea and store it in the database.

3.3 Genetic markers (loci)

MEGAPTERA is not a super-matrix approach *sensu* Smith *et al.* (2009), where sequences are sorted by $N \times N$ comparisons into orthologous groups, but instead requieres the specification of genetic marker regions (here after called 'loci'). Unfortunately there is no strict definition of how loci should be called (and in which fields they should be searched for, section 4) at NCBI. The widely used large unit of ribosomal DNA, e.g., has been referred to by 28S, 26S, or 25S according to its different molecular weight in different lineages. Thus, MEGAPTERA offers (and you should use it) the possibility to define aliases of locus names.

The order of aliases has no effect, but you might prefer a short one as the first argument is taken to derive several internally used labels for database tables and columns.

Let's choose five loci that are widely used in vertebrate phylogenetics.

```
loci <- list(locus("cytB", "Cytb"),
             locus("cox1", "COI"),
             locus("16S"),
             locus("RAG1"),
             locus("RAG2")
)

##
## Checking if locus exists on GenBank ..
## .. found 333617 records
## Checking if locus exists on GenBank ..
## .. found 1078537 records
## Checking if locus exists on GenBank ..
## .. found 7771830 records
## Checking if locus exists on GenBank ..
## .. found 34160 records
## Checking if locus exists on GenBank ..
## .. found 12118 records
```

3.4 Pipeline parameters

The pipeline requires a set of parameters that can be set with `megapteraPars`; see `help("megapteraPars-class")` for a detailed description of each parameter.

```
pars <- megapteraPars()
pars

## MEGAPTERA connection parameters:
##            update.seqs = all
##        max.gi.per.spec = 100
##                 max.bp = 5000
##   reference.stringence = 0
##     reference.max.dist = 0.25
##     min.seqs.reference = 10
##               max.dist = 0.55
##             fract.miss = 0.25
##                filter1 = 0.5
##                filter2 = 0.25
##                filter3 = 0.05
##                filter4 = 0.2
##         block.max.dist = 0.5
##              min.n.seq = 5
##                    gb1 = 0.5
##                    gb2 = 0.5
##                    gb3 = 9999
##                    gb4 = 2
##                    gb5 = a
```

3.5 Collect input data

Now we can bundle all input data in one single object of class "megapteraProj". The arguments db, taxon, and pars are given objects of classes "dbPars", "taxon", and "megapteraPars", respectively. Note that we leave argument locus undefined; this argument will be set later on (4). The input data will be completed by giving the paths to the alignment and masking programs .

```
x <- megapteraProj(db = conn,
                   taxon = tax,
                   locus = locus(),
                   align.exe = "/usr/local/bin/mafft",
                   mask.exe = "/Applications/Gblocks_0.91b")
x

## --- megaptera project data ---
## ingroup taxon  : Cetacea
## outgroup taxon : Hippopotamus amphibius, Sus scrofa , ... [ 3 ]
## in kingdom     : Metazoa
## hybrids        : excluded
## locus          : undefined
## execution      : serial
## update         : no
```

4 Running the pipeline

Finally we are ready for running the pipeline. The eight steps of the pipeline will be run separately for each locus by looping over the list of loci defined in 3.3. In each iteration, setLocus is used to set the locus argument accordingly.

```
for ( i in loci) {
  x <- setLocus(x, i)
  stepA(x)
  stepB(x)
  stepC(x)
  stepD(x)
  stepE(x)
  stepF(x)
  stepG(x)
  stepH(x)
}
```

5 Concatenation of loci

```
conn <- dbConnect(PostgreSQL(), port = 5432, user = "postgres",
                  dbname = "cetacea", password = "oxalis")
o <- c("ord", "-")
s <- c("ord", "Cetacea")
check.Markers(conn, outgroup = o, subset = s)
sm <- supermatrix(conn, outgroup = o, subset = s)
dbDisconnect(conn)
sm
```

6 References

Smith, S. A., Beaulieu, J. M. & Donoghue, M. J. (2009) Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. *BMC Evolutionary Biology*, **9**, 37. doi:10.1186/1471-2148-9-37.