Managing big biological sequence data with *Biostrings* and *DECIPHER*

Erik Wright
University of Wisconsin-Madison
What you should learn

- How to use the **Biostrings** and **DECIPHER** packages
- Creating a database to store sequences
- Adding data to the database
- Querying for specific sequences in the database
- Manipulating **XStringSet** objects
- Run large-scale analyses in pieces
R packages for biological seqs.

**Biostrings**
- editing sequences
- subsequence
- masking
- reverse complement
- oligo frequencies
- searching
- translation

**DECIPHER**
- consensus
- restriction digest
- alignment
- syntenic mapping
- sequence databases
- primer/probe design
- phylogenetic trees

**XStringSets**
- sequence
- databases
Coverage in this workshop

Biostrings
- oligo frequencies
- searching
- translation
- reverse complement
- subsequence
- editing sequences

DECIPHER
- consensus
- restriction digest
- alignment
- primer/probe design
- sequence databases
- phylogenetic trees

XStringSets
Put on your detective hat...

Pennsylvania

Pythium

Soybean fields
Identifying *Pythium* species

**Taxonomy:**
Eukaryota
Chromalveolata
Heterokontophyta
Oomycota
Pythiales
Pythiaceae
Pythium

Mitochondrial genome

Cytochrome c oxidase subunit 1 (COI gene)

Nuclear genome

Let's get started!

# first it is necessary to get the datasets used in this tutorial
# the datasets are located in the BigBioSeqData package
# normally we would simply use library(DECIPHERER)

> library(BigBioSeqData)

> help(package="BigBioSeqData")

# click the link for "User guides, package vignettes and other documentation"
Overview of workflow

• Part 1:
  - Import publicly available sequences into a database
  - Design primers targeting *Pythium* COI gene
  - (Wet lab work: amplify DNA, sequence)
• Part 2:
  - Import the new amplicon sequences
  - Quality trim the sequences
  - Cluster the *Pythium* sequences into groups
• Part 3:
  - Align the cluster representatives to sequences from known species
  - Identify the *Pythium* strains present in each sample
Overview of workflow part #1

sequence repository

download Pythium COI sequences

import into seq. database

align the sequences

cluster into groups

design primers
Seqs2DB function

# Import sequences from a GenBank formatted file
Seqs2DB(paste(data_dir,
    "/Pythium_spp_COI.gb",
    sep=""),
    type="GenBank",
    dbFile=dbConn,
    identifier="Pythium")

Arguments (in order):

1. seqs = XStringSet or path to text file .gz, .bzip2, .xz also supported http:// and ftp:// supported
2. type = "GenBank", "FASTQ", "FASTA" or "XStringSet"
3. dbFile = Database connection or path to SQLite database file
4. identifier = character string uniquely identifying this batch of sequences
Creating a sequence database

```
# Import sequences from a GenBank formatted file
Seqs2DB(paste(data_dir, 
    "/Pythium_spp_COI.gb", 
    sep=""),
    type="GenBank",
    dbFile=dbConn,
    identifier="Pythium")
```
Viewing a database table

# View the database table that was constructed
BrowseDB(dbConn)
Retrieving sequences

```r
# Retrieve the imported sequences
> dna <- SearchDB(dbConn)
Search Expression:
select row_names, sequence from _Seqs where
row_names in (select row_names from Seqs)

DNAStringSet of length: 488
Time difference of 0.03 secs

> dna
    width seq                     names
[1]  1277 ATGAATTTT...GTTATTCTT 1
[2]  1277 ATGAATTTT...GTTATTTTT 2
[3]  1095 TATATAATG...TATTTTTTT 3
[4]  1299 ATGAATTTT...ATTACATTT 4
[5]  1109 CATCATTTA...TATAGGTGT 5
...   ...     ...
[484]  673 AAATCATAA...TTATTCCAA 484
[485]  680 AATCATAAA...ACATTTTATT 485
[486]  680 AATCATAAA...ACATTTTATT 486
[487]  680 AATCATAAA...ACATTTTATT 487
[488]  680 AATCATAAA...ACATTTTATT 488
```

Features of SearchDB:

1. Automatically builds a database query
2. Displays the query if `verbose=TRUE` (default)
3. Auto-detects the type of sequences to return (DNA, RNA, or AAStringSet)
SearchDB: optional arguments

SearchDB(dbFile,
    tblName = "Seqs",
    identifier = "",
    type = "XStringSet",
    limit = -1,
    replaceChar = "-",
    nameBy = "row_names",
    orderBy = "row_names",
    countOnly = FALSE,
    removeGaps = "none",
    clause = "",
    processors = 1,
    verbose = TRUE)

Choose which table to query
Constrain to a subset of identifiers in the table
Detect (X) the sequence type, or specify (DNA/RNA/AA/B)
Limit the number of sequences
Replace unsupported letters with another (e.g., "-"
Name and order the seqs. according to the values in these database columns
Return the number of seqs.
Remove gaps from sequences if they are aligned
Append a clause to the query
Decompress using \textit{n} cores
Multiple sequence alignment

AlignSeqs(seqs)

DNA or RNA or AA

Aligned sequences

AlignTranslation(dna)

Coding region

Align amino acids

Aligned DNA

AlignDB(dbConn, tblName = c("Seqs1", "Seqs2"))

Merged alignment

DesignProbes function

DesignSignatures(dbConn, type = "sequence")

HRM or FLP or Sequencing

± restriction digestion

PCR or qPCR

Target group

Non-target group

Overview of workflow part #2

perform amplicon sequencing

obtain COI sequences

import into new table

trim by quality scores

identify potential Pythium sequences

cluster Pythium sequences

Trimming sequences by quality
Performing analyses in parts

The key idea: process batches of sequences separately
- Use the "offset,limit" feature in queries

```
> nSeqs <- SearchDB(dbConn, count = TRUE, verbose = FALSE)
> offset <- 0
> while (offset < nSeqs) {
    dna <- SearchDB(dbConn,
        limit = paste(offset, 1e4, sep = "","),
        verbose = FALSE)
    # do something with dna
    offset <- offset + 1e4
}
```

offset,limit:
"0,1e4"
"1e4,1e4"
"2e4,1e4"
...
Performing analyses in parts

The key idea: process batches of sequences separately
- Use the "offset,limit" feature in queries
- Select sequences belonging to each identifier

```r
> ids <- dbGetQuery(dbConn, "select distinct identifier from Reads")
> for (i in seq_along(ids$identifier)) {
  dna <- SearchDB(dbConn,
      identifier = ids$identifier[i],
      verbose = FALSE)

  # do something with dna
}
```
Overview of workflow part #3

- choose species representatives
- select cluster representatives
- align combined sequences
- construct a distance matrix
- build a neighbor joining tree
- identify known *Pythium* species