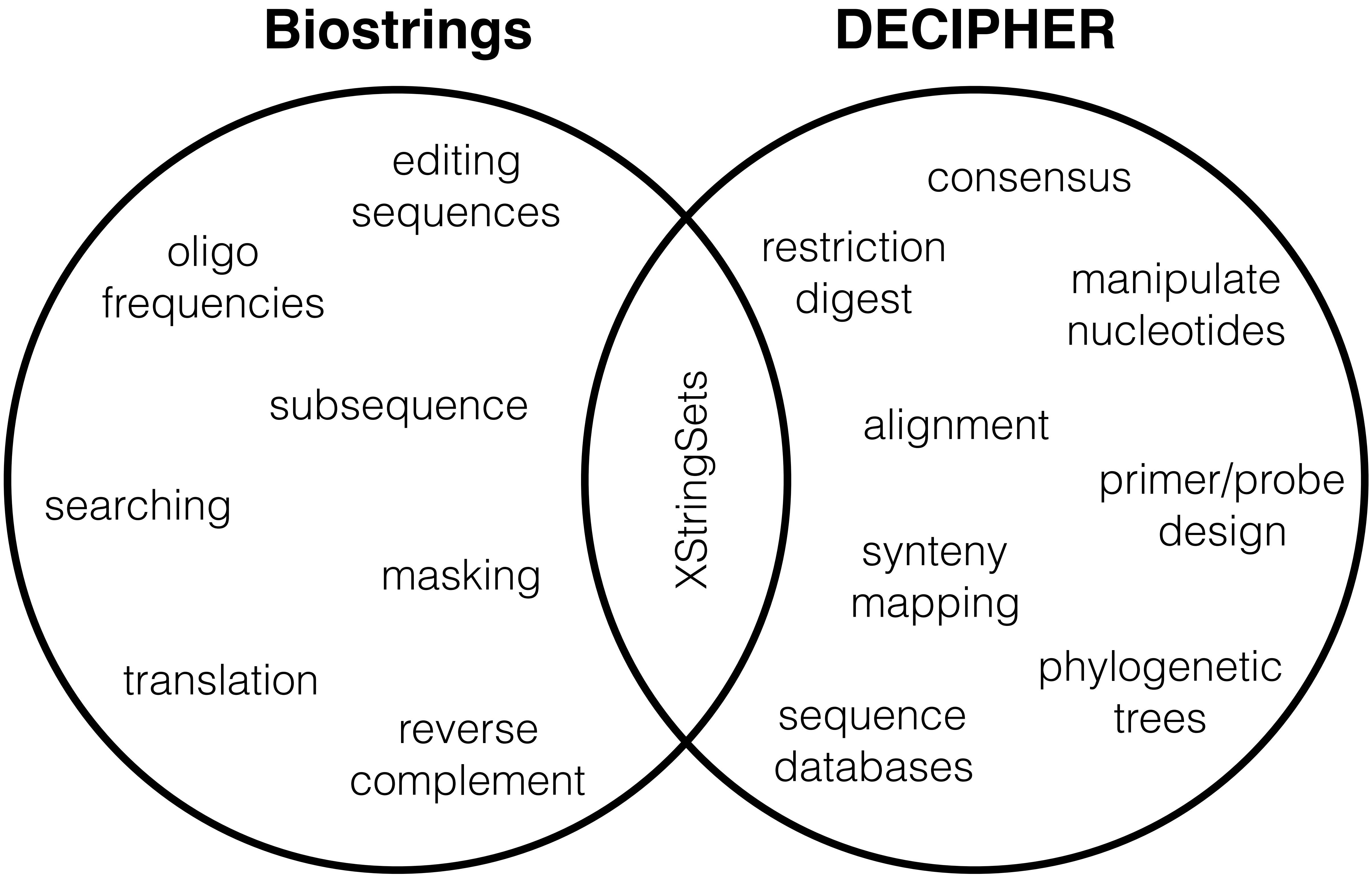
Managing big biological sequence data with *Biostrings* and *DECIPHER*

Erik Wright
University of Wisconsin-Madison

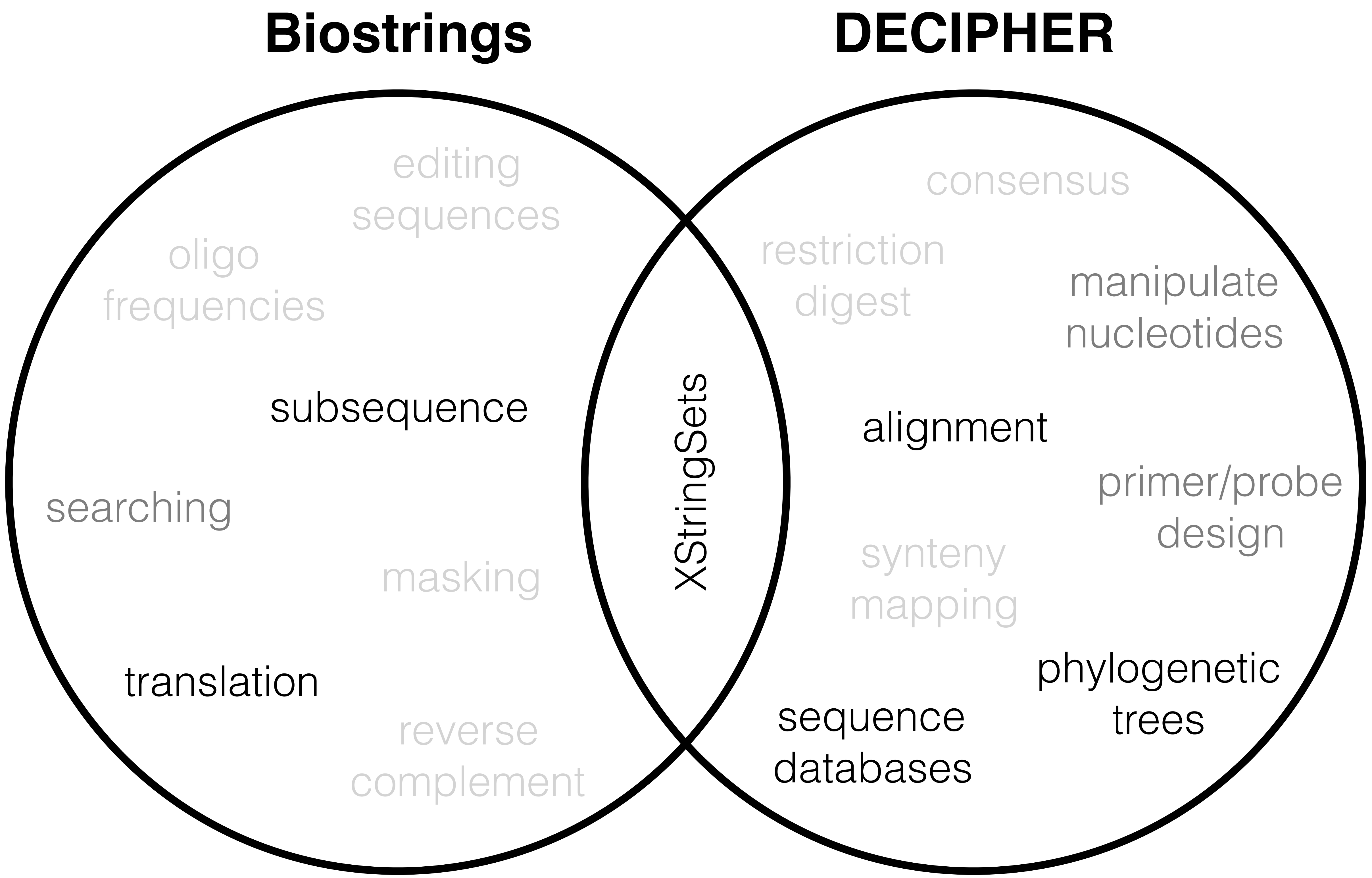
What you should learn

- 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

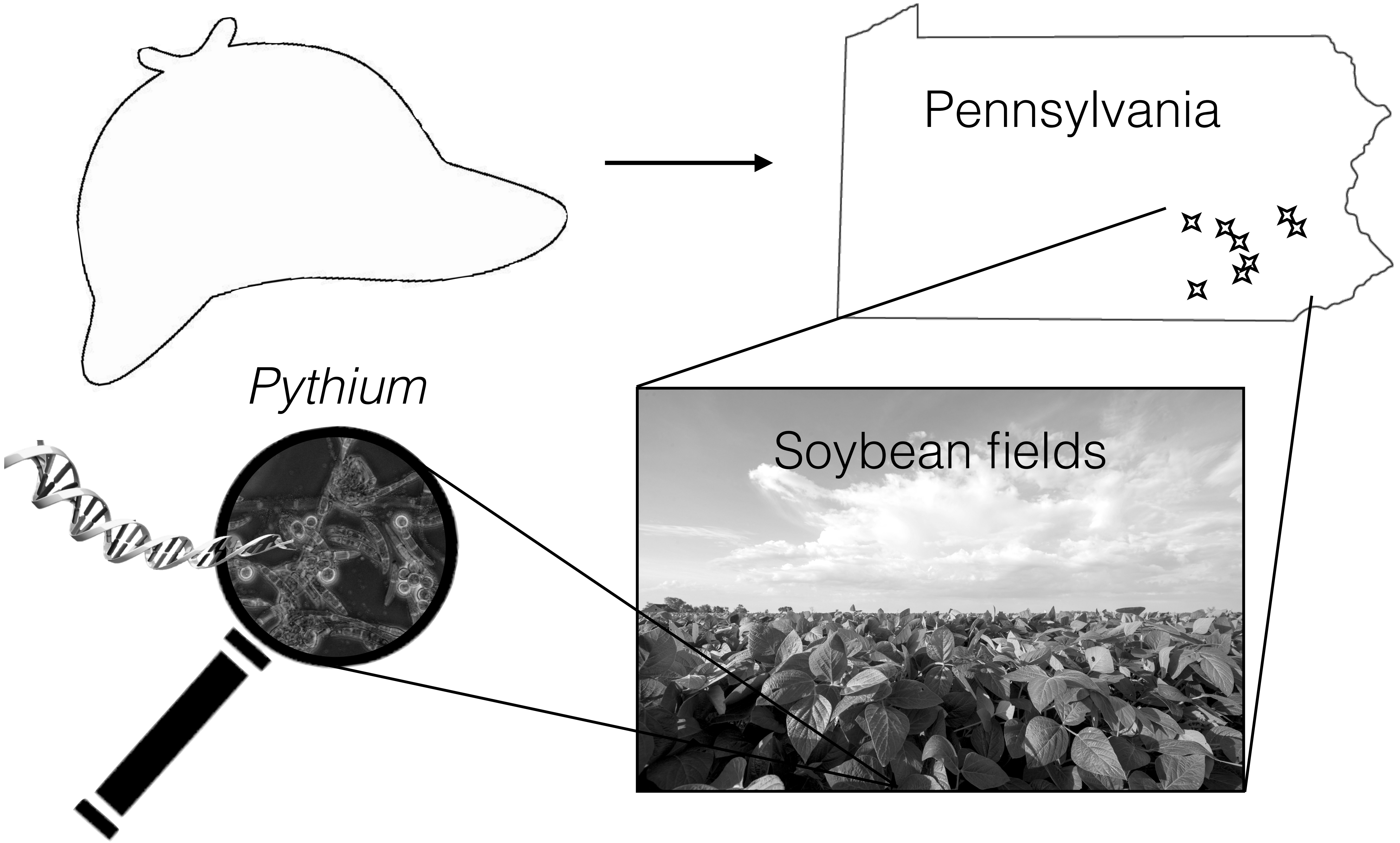
R packages for biological seqs.



Coverage in this workshop



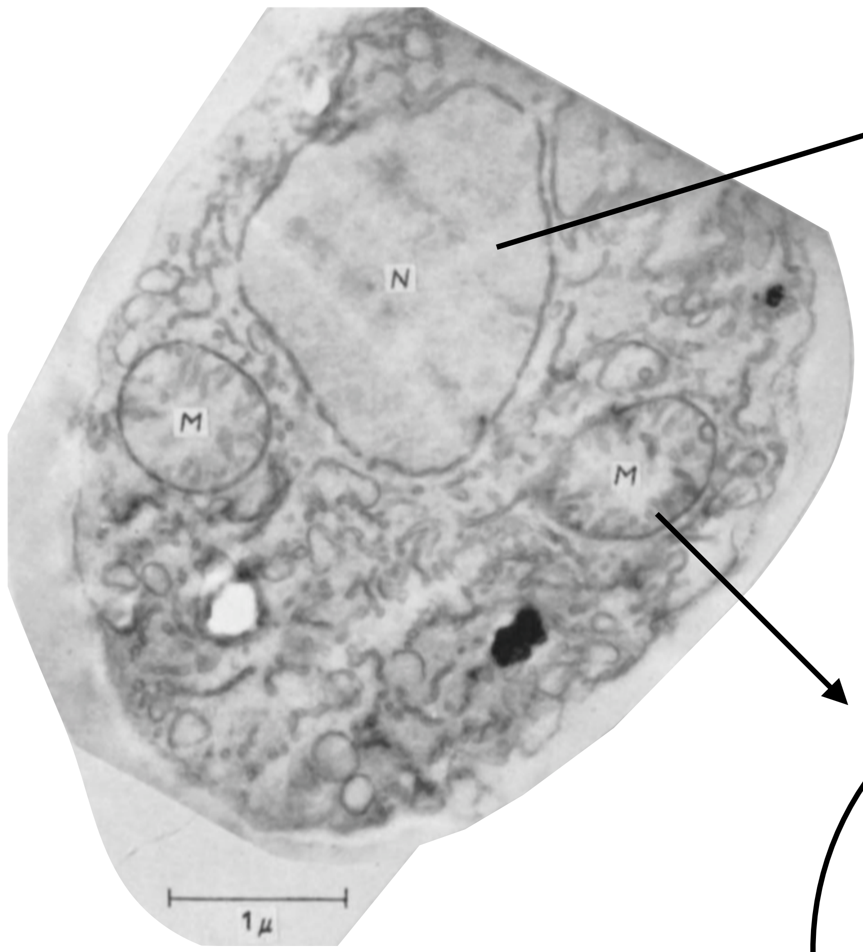
Put on your detective hat...



Identifying *Pythium* species

<u>Taxonomy:</u>

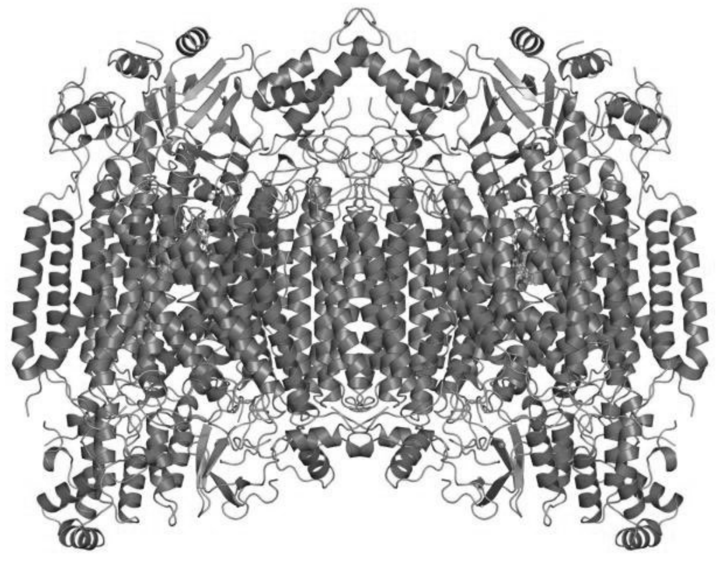
- Eukaryota
  - Chromalveolata
    - Heterokontophyta
      - Oomycota
        - Pythiales
          - Pythiaceae
            - Pythium



Nuclear genome

Mitochondrial genome

Cytochrome c oxidase subunit 1 (**COI gene**)



Hawker, L. & Abbott, P. (1963). Journal Gen. Microbiol.

Let's get started!

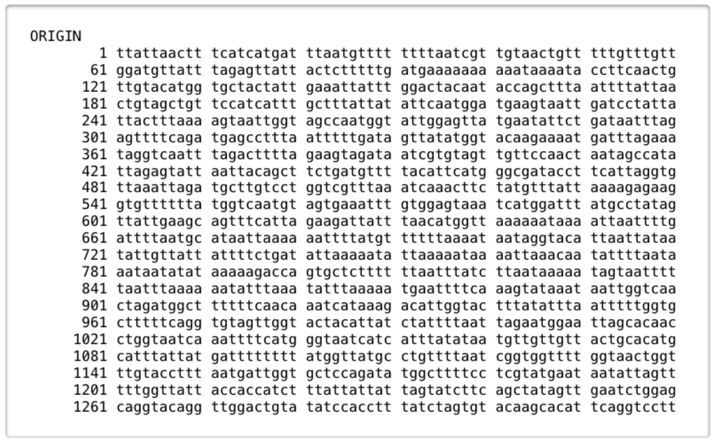
```
# first it is necessary to get the datasets used in this
tutorial
# the datasets are located in the BigBioSeqData
package
# normally we would simply use library(DECIPHER)

> 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

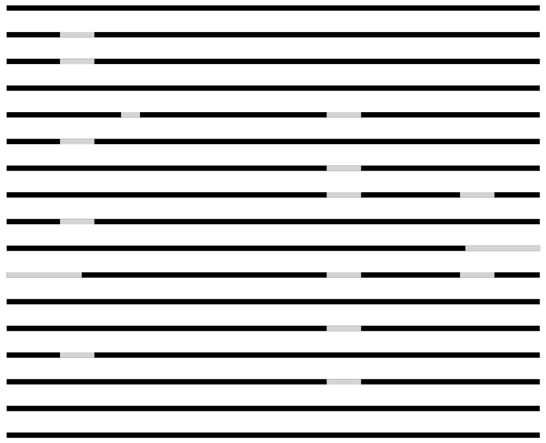
sequence repository



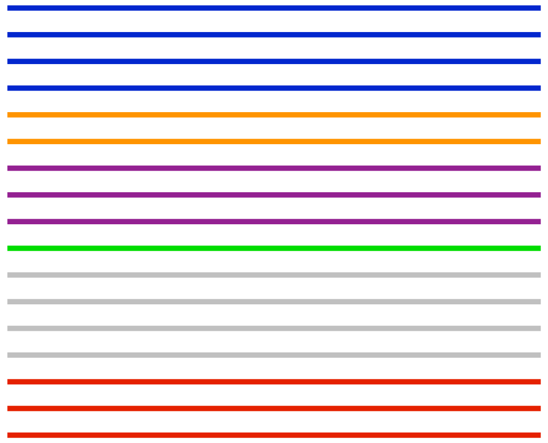
```
ORIGIN
        1 ttattaactt tcatcatgat ttaatgtttt ttttaatcgt tgtaactgtt tttgtttgtt
       61 ggatgttatt tagagttatt actctttttg atgaaaaaaa aaataaaata ccttcaactg
      121 ttgtacatgg tgctactatt gaaattattt ggactacaat accagcttta attttattaa
      181 ctgtagctgt tccatcattt gctttattat attcaatgga tgaagtaatt gatcctatta
      241 ttactttaaa agtaattggt agccaatggt attggagtta tgaatattct gataatttag
      301 agttttcaga tgagccttta atttttgata gttatatggt acaagaaaat gatttagaaa
      361 taggtcaatt tagactttta gaagtagata atcgtgtagt tgttccaact aatagccata
      421 ttagagtatt aattacagct tctgatgttt tacattcatg ggcgatacct tcattaggtg
      481 ttaaattaga tgcttgtcct ggtcgtttaa atcaaacttc tatgtttatt aaaagagaag
      541 gtgtttttta tggtcaatgt agtgaaattt gtggagtaaa tcatggattt atgcctatag
      601 ttattgaagc agtttcatta gaagattatt taacatggtt aaaaaataaa attaattttg
      661 attttaatgc ataattaaaa aattttatgt tttttaaaat aataggtaca ttaattataa
      721 tattgttatt attttctgat attaaaaata ttaaaaataa aattaaacaa tattttaata
      781 aataatatat aaaaagacca gtgctctttt ttaatttatc ttaataaaaa tagtaatttt
      841 taatttaaaa aatatttaaa tatttaaaaa tgaattttca aagtataaat aattggtcaa
      901 ctagatggct tttttcaaca aatcataaag acattggtac tttatattta atttttggtg
      961 ctttttcagg tgtagttggt actacattat ctattttaat tagaatggaa ttagcacaac
     1021 ctggtaatca aattttcatg ggtaatcatc atttatataa tgttgttgtt actgcacatg
     1081 catttattat gatttttttt atggttatgc ctgttttaat cggtggtttt ggtaactggt
     1141 ttgtaccttt aatgattggt gctccagata tggcttttcc tcgtatgaat aatattagtt
     1201 tttggttatt accaccatct ttattattat tagtatcttc agctatagtt gaatctggag
     1261 caggtacagg ttggactgta tatccacctt tatctagtgt acaagcacat tcaggtcctt
```

download *Pythium* COI sequences

import into seq. database



align the sequences



cluster into groups

design primers

amplicon sequencing (part #2)

Seqs2DB function

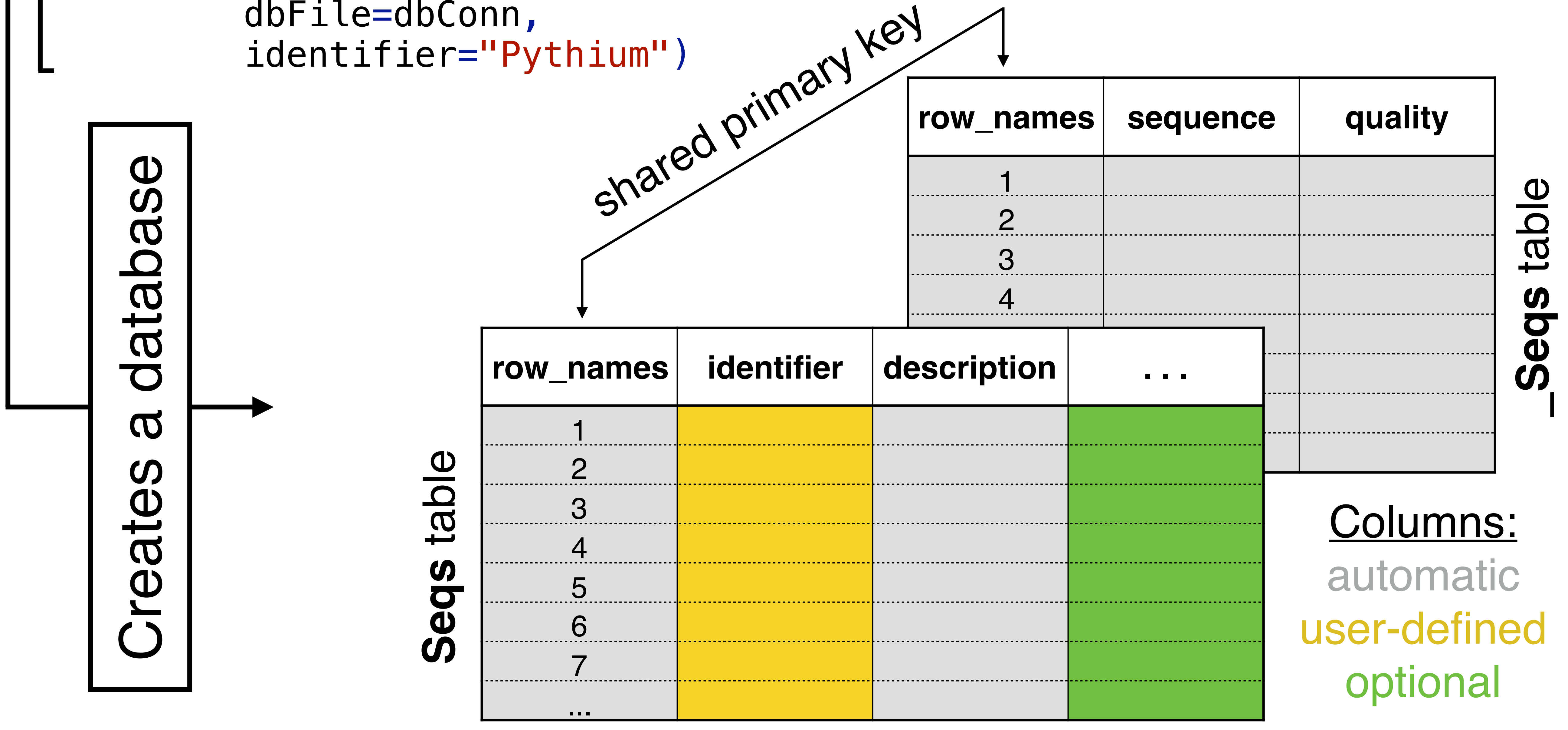
```r
# Import sequences from a GenBank formatted file
Seqs2DB(paste(data_dir,
              "/Pythium_spp_COI.gb",
              sep=""),
        type="GenBank",
        dbFile=dbConn,
        identifier="Pythium")
```

<u>Arguments (in order):</u>

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")
```

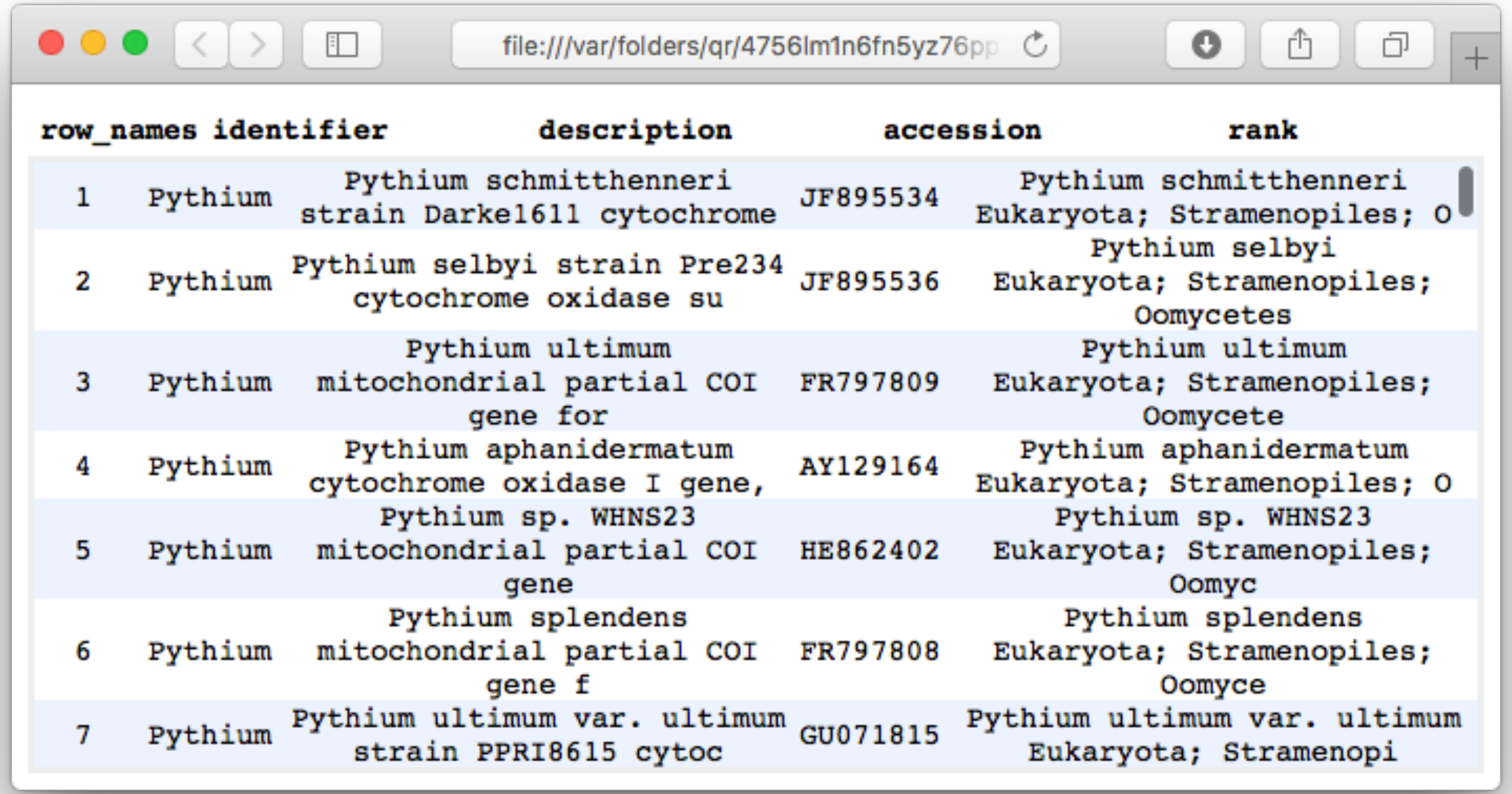


Wright, E. (2016). The R Journal.

Viewing a database table

```r
# View the database table that was constructed
BrowseDB(dbConn)
```

Displays a database



file:///var/folders/qr/4756lm1n6fn5yz76pp

row_names	identifier	description	accession	rank
1	Pythium	Pythium schmitthenneri strain Darkel611 cytochrome	JF895534	Pythium schmitthenneri Eukaryota; Stramenopiles; O
2	Pythium	Pythium selbyi strain Pre234 cytochrome oxidase su	JF895536	Pythium selbyi Eukaryota; Stramenopiles; Oomycetes
3	Pythium	Pythium ultimum mitochondrial partial COI gene for	FR797809	Pythium ultimum Eukaryota; Stramenopiles; Oomycete
4	Pythium	Pythium aphanidermatum cytochrome oxidase I gene,	AY129164	Pythium aphanidermatum Eukaryota; Stramenopiles; O
5	Pythium	Pythium sp. WHNS23 mitochondrial partial COI gene	HE862402	Pythium sp. WHNS23 Eukaryota; Stramenopiles; Oomyc
6	Pythium	Pythium splendens mitochondrial partial COI gene f	FR797808	Pythium splendens Eukaryota; Stramenopiles; Oomyce
7	Pythium	Pythium ultimum var. ultimum strain PPRI8615 cytoc	GU071815	Pythium ultimum var. ultimum Eukaryota; Stramenopi

Retrieving sequences

```
# 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
  A DNAStringSet instance of length 488
      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...ACATTTATT 485
[486]   680 AATCATAAA...ACATTTATT 486
[487]   680 AATCATAAA...ACATTTATT 487
[488]   680 AATCATAAA...ACATTTATT 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.

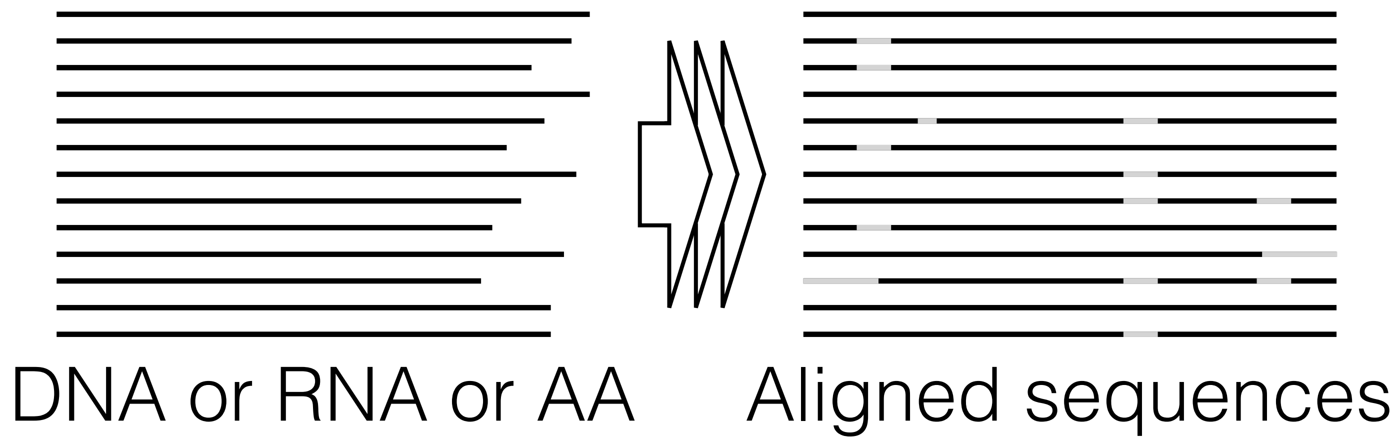
Remove gaps from sequences if they are aligned

Append a clause to the query

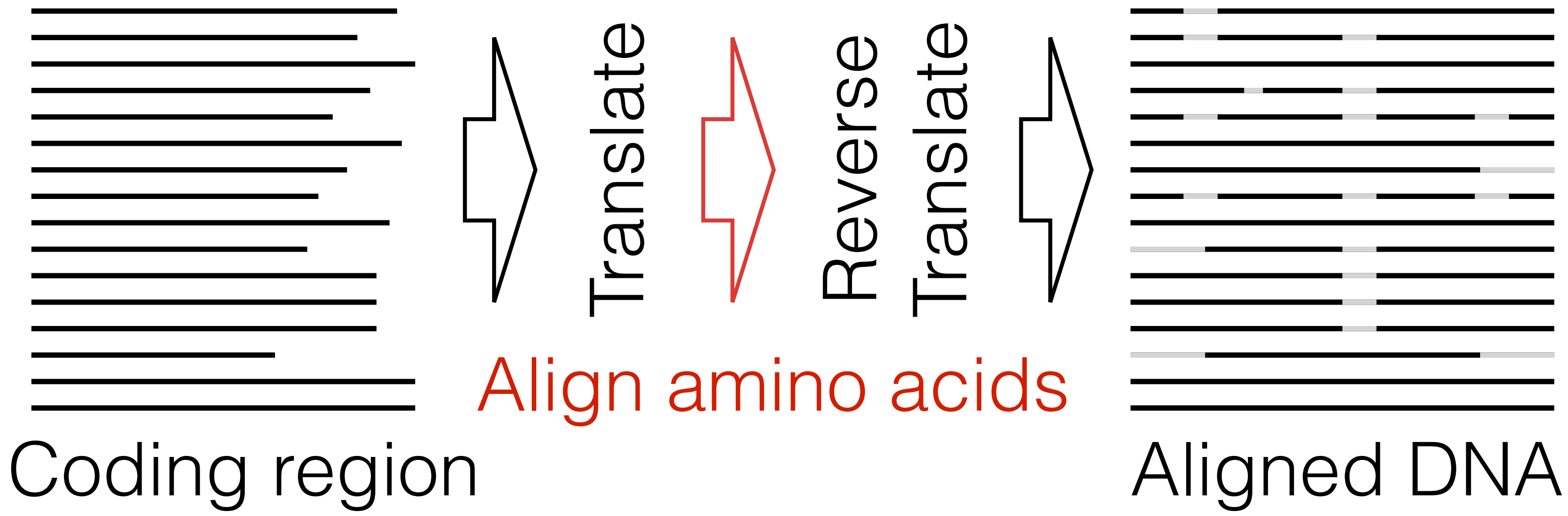
Decompress using *n* cores

Multiple sequence alignment

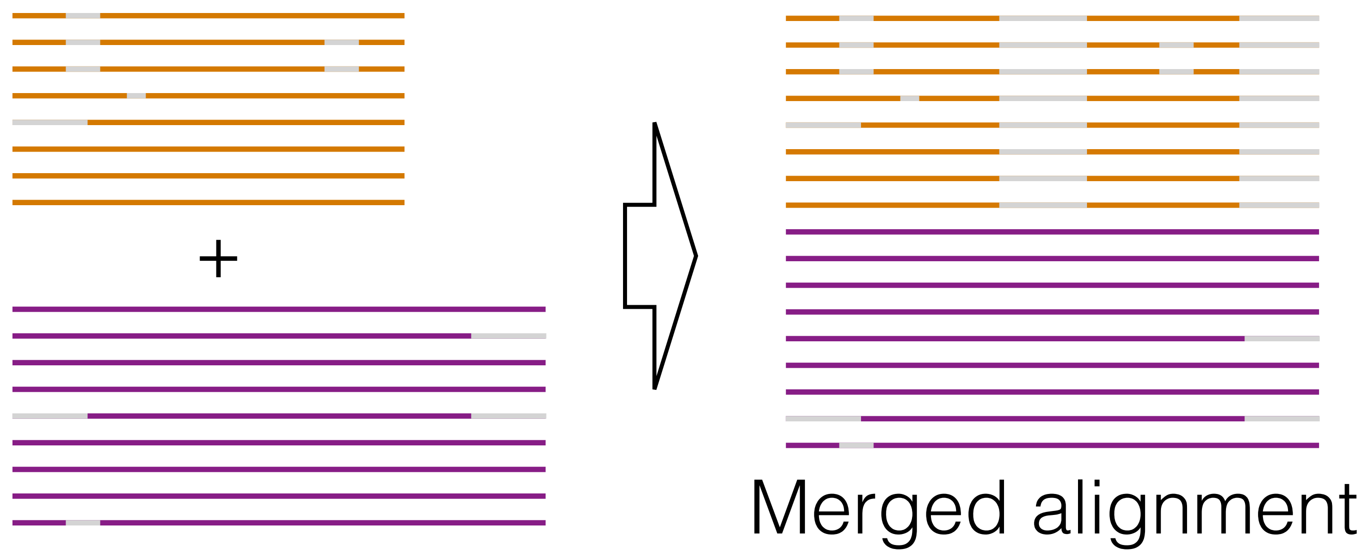
```
AlignSeqs(seqs)
```



```
AlignTranslation(dna)
```



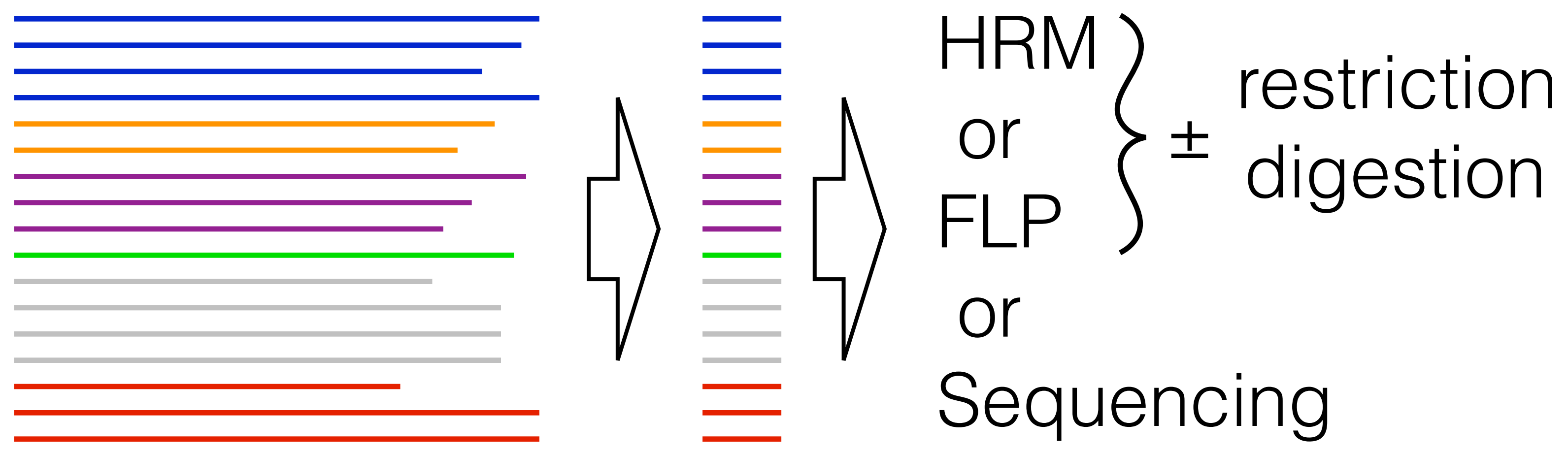
```
AlignDB(dbConn,
    tblName = c("Seqs1",
                "Seqs2"))
```



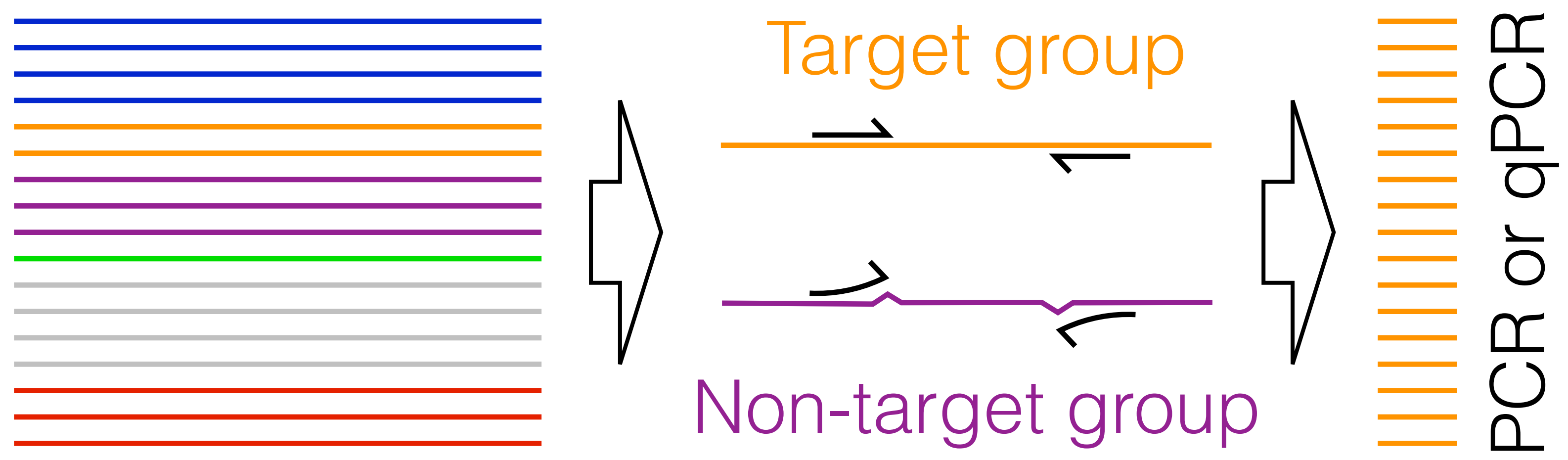
Wright, E. (2015). BMC Bioinformatics.

DesignProbes function

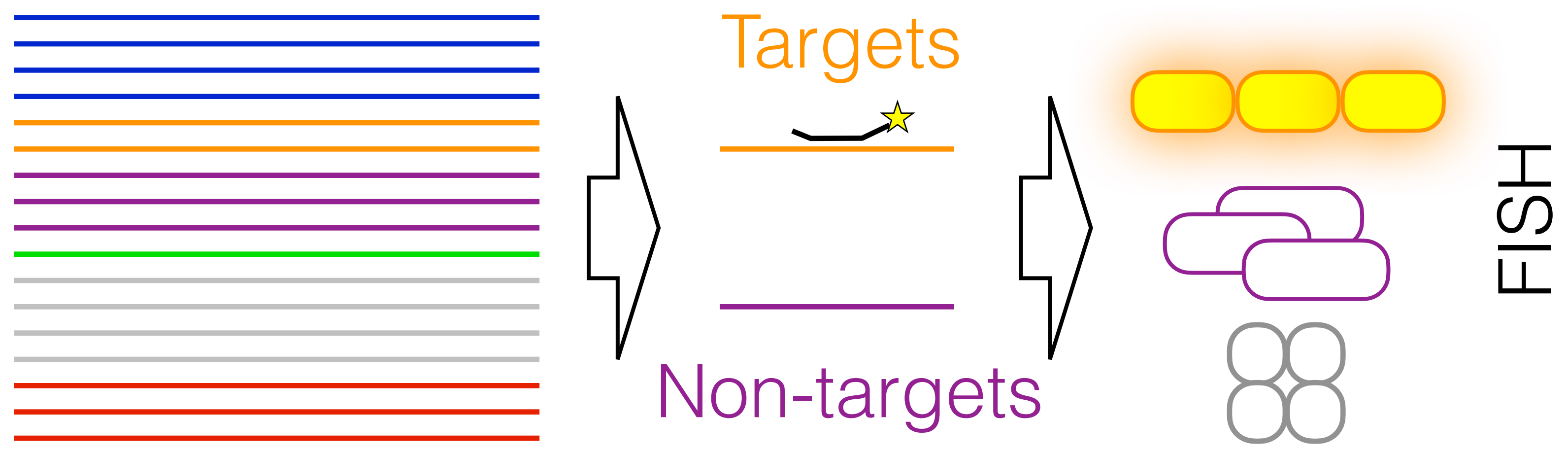
```
DesignSignatures(dbConn,
      type = "sequence")
```



```
tiles <- TileSeqs(dbConn)
DesignPrimers(tiles,
      numPrimerSets = 10)
```

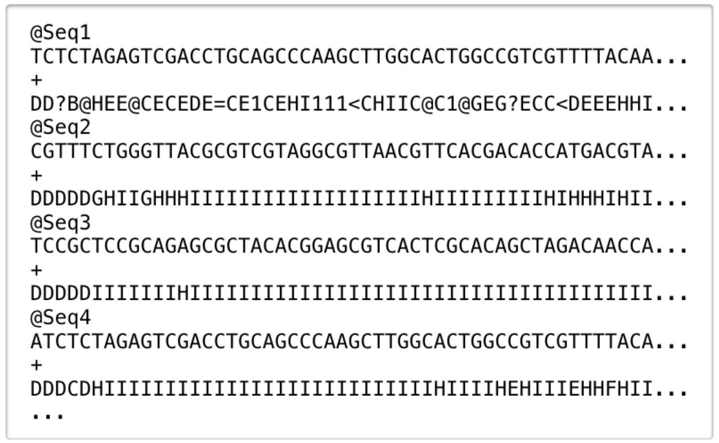


```
tiles <- TileSeqs(dbConn)
DesignProbes(tiles,
       numProbeSets = 10)
```

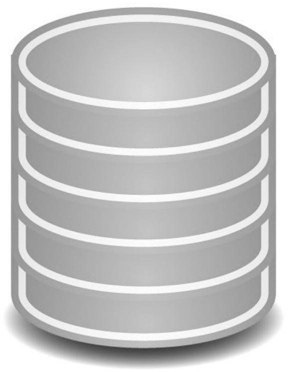


Wright, E. *et al.* (2014). Applied and Environmental Microbiology.

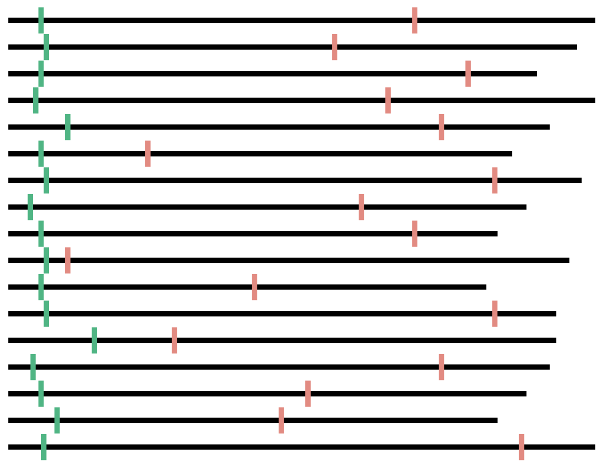
Overview of workflow part #2

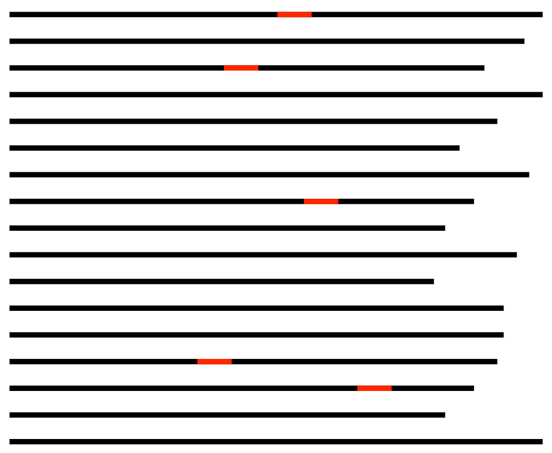


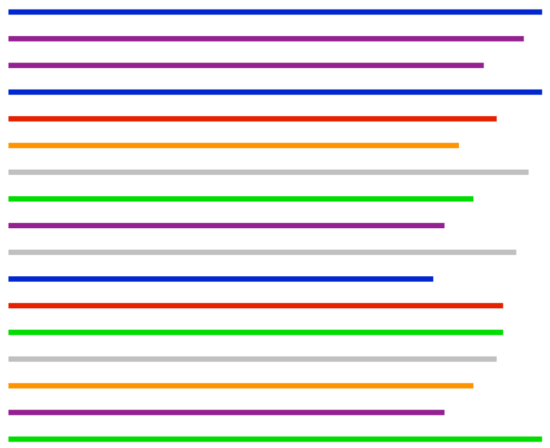
```
@Seq1
TCTCTAGAGTCGACCTGCAGCCCAAGCTTGGCACTGGCCGTCGTTTTACAA...
+
DD?B@HEE@CECEDE=CE1CEHI111<CHIIC@C1@GEG?ECC<DEEEHHI...
@Seq2
CGTTTCTGGGTTACGCGTCGTAGGCGTTAACGTTCACGACACCATGACGTA...
+
DDDDDGHIIGHHHIIIIIIIIIIIIIIIIIIHIIIIIIIIIIHIHHHIHII...
@Seq3
TCCGCTCCGCAGAGCGCTACACGGAGCGTCACTCGCACAGCTAGACAACCA...
+
DDDDDIIIIIIHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII...
@Seq4
ATCTCTAGAGTCGACCTGCAGCCCAAGCTTGGCACTGGCCGTCGTTTTACA...
+
DDDCDHIIIIIIIIIIIIIIIIIIIIIIIIIIIHIIIIHEHIIIEHHFHII...
...
```



perform amplicon sequencing → obtain COI sequences → import into new table →



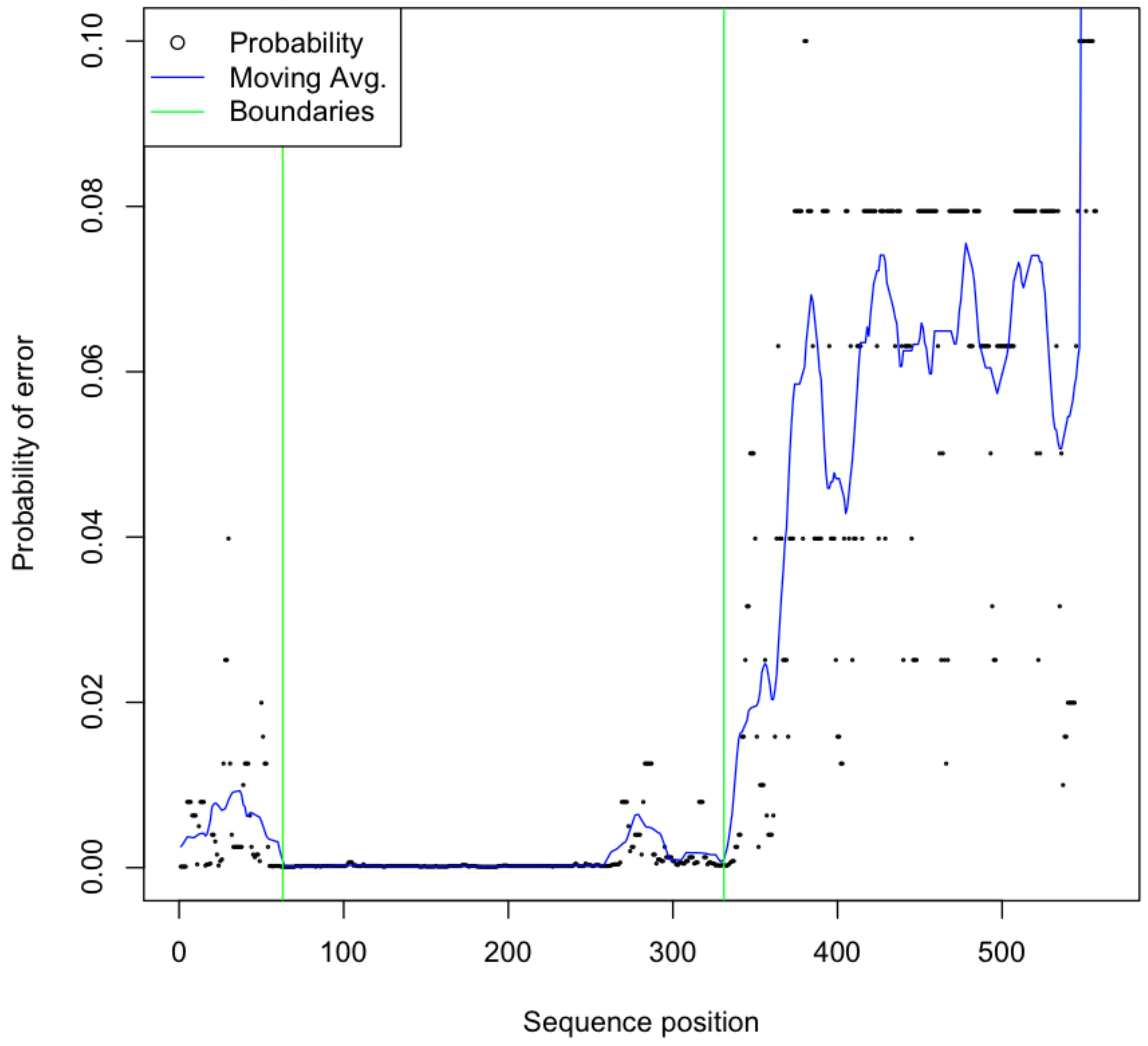




trim by quality scores → identify potential *Pythium* sequences → cluster *Pythium* sequences

Dataset: Coffua, L., *et al.* (2016). Plant Disease.

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"
...

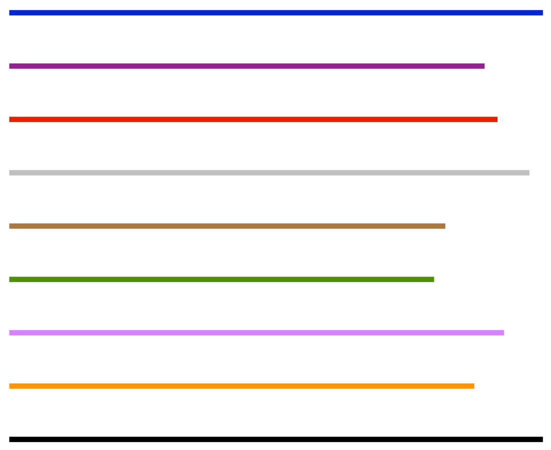
Performing analyses in parts

The key idea: process batches of sequences separately

- Use the "offset,limit" feature in queries
- Select sequences belonging to each identifier

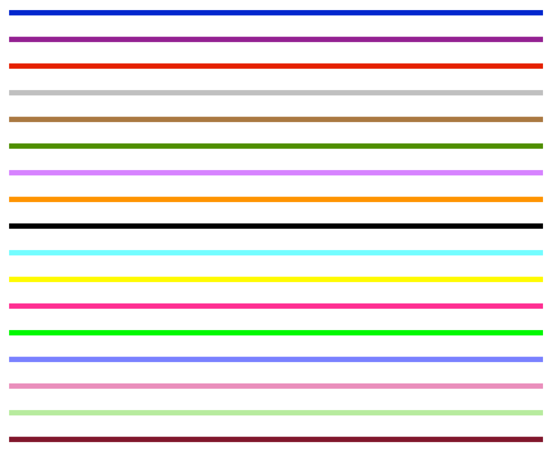
```
> ids <- dbGetQuery(dbConn, "select distinct identifier from Reads")
> for (i in seq_along(ids$identifier)) {
     dna <- SearchDB(dbConn,
         identifier = ids$identifier[i],
         verbose = FALSE)

     # do something with dna
  }
```

Overview of workflow part #3

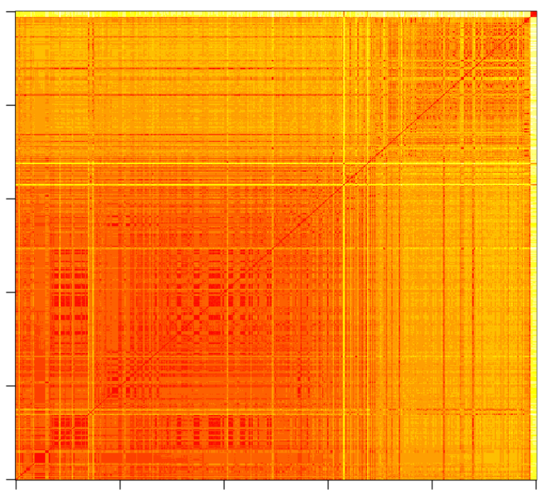


choose species representatives

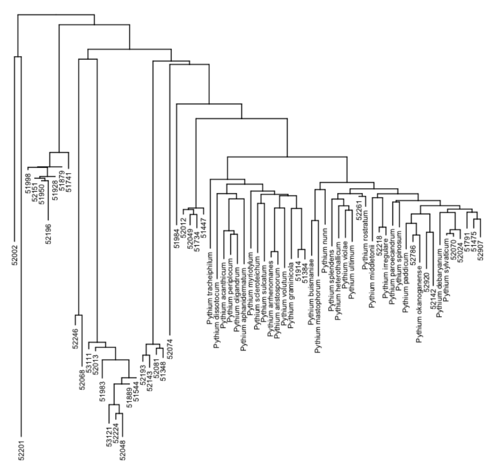
select cluster representatives



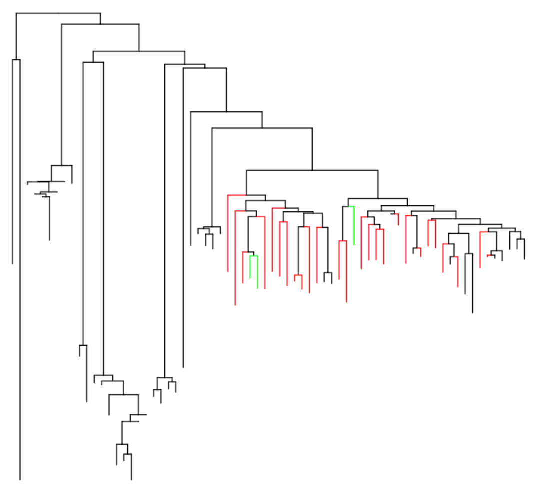
align combined sequences



construct a distance matrix



build a neighbor joining tree



identify known *Pythium* species