Introduction to Programming in R: alignment

Erik Spence

SciNet HPC Consortium

19 April 2023

Today's slides

Today's slides can be found here. Go to the "Introduction to programming in R" page, under Lectures, "Alignment".

https://scinet.courses/1277

Today's Lecture

Today's class will explore the wild wild world of file input and output in R.

- Introduction to alignment.
- BLAST.
- rBLAST.
- Example.

What is alignment?

Alignment is a matching technique for finding similar sequences. It allows for imperfections in sequences, like substitutions, omissions and gaps.

Typically, we match shorter sequences ('queries') to longer 'target' (or 'reference') sequences, or to a 'search set' or database of target sequences.

We can do this for nucleotide as well as protein sequences.

We usually distinguish between 'global alignment' (matching everything) and 'local alignment' (matching substrings).

```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
             ||||  ||||||  ||||||||||||||||
          5' TACTCACGGATGAGGTACTTTAGAGGC 3'
```

What is alignment used for?

Alignment is used to examine a number of different problems:

- Identifying sequences from your lab with known reference genomes.
- Finding genes from one organism in the DNA of other species.
- Finding conserved sequences and motifs across species.
- Finding evolutionary relationships.
- Finding structural or functional relationships between proteins.
- And many others.

Sequence alignment shows up in other fields as well, such as text analysis.

Why not use regexes?

A technique known as "regular expressions" (regex) is commonly used for pattern matching, especially on Linux systems.

These do not work well for local alignment, because:

- Allowing for gaps to be anywhere will match almost anything.
- Regexes do not have any notion of how well the pattern matches.
- This makes it hard to put in biological aspects (e.g., substituting T for C is not as bad as A for C, longer gaps are more likely than a lot of small gaps, *etc.*)
- Regexes look at all possible matches, which is inefficient.

We will use other techniques instead.

Alignment in a nutshell

What is the goal, exactly? Given some 'target' sequence, such as `CTGA...AGTTAGTGG...`

and some 'query' sequence, such as `AGTTCCCTG`.

One possible alignment is:

```
CTGA...AGTTAG--TGG...
      ||||    ||
      AGTT*CCCTG
```

Note that 'gaps' and substitutions' are allowed.

The quality of the alignment is assessed using a 'scoring criteria'. Scoring criteria can be rather sophisticated.

Alignment in a nutshell, continued

Ok, but how is alignment actually done?

- The so-called Smith-Waterman algorithm lies at the heart of most local alignment algorithms, and is very efficient (much better than brute force matching).
- The algorithm assigns a 'score'. Roughly speaking, the fewer the matches between individual nucleotides, the lower the score.
- Instead of raw scores, alignment tools often report the likelihood that the match is due to chance, the 'e-value'.
- We are almost always interested in the 'highest scoring segment pairs' (HSPs), *i.e.*, the ones with the lower e-values.

Optimized but heuristic alignment

Straight-up Smith-Waterman is good, but can be improved even more.

- Although gaps are allowed, HSPs will have substantial stretches of exact matches.
- This realization is the basis of more efficient algorithms.
- Such algorithms first search for short exact matches of length k, then do alignment around those spots.
- The search for short exact matches is done using a 'hash', storing locations of all short sequences of length k in the query. Some aligners also hash k-mers in the target sequence.
- This is the 'heuristic' approach; it cannot guarantee that the best match is found.
- The size of the exact match is a parameter. $k \approx 11$ for DNA, $k \approx 4$ for proteins.

While searching around the exact matches doesn't guarantee that you'll find the best match, it makes it significantly more efficient, and the odds of missing the best match are low.

BLAST

One of the most-commonly-used alignment tools is known as BLAST (Basic Local Alignment Search Tool). It is a suite of programs that find alignments in a series of sequences using this heuristic approach.

Advantages:

- No need to code your own aligner.
- It's a fast algorithm.
- It's implemented in C++: fast!
- It can be run over the web.
- It can also run on own computer or a supercomputer.
- It's being actively developed: BLAST+ > BLAST2 > BLAST1

`https://blast.ncbi.nlm.nih.gov/Blast.cgi`

Performing BLAST queries

There are several ways to run BLAST queries.

- Run online though a browser:
  - blast.ncbi.nlm.nih.gov/Blast.cgi
  - Convenient, point and click,
  - Not codable, cannot be automated, and so not easily reproducible,
  - Subject to internet connectivity,
  - Cannot use your own sequence database.
- Run locally using local targets.
  - Using the rBLAST packages we can do queries locally from within R,
  - Codable, automatable.
  - You can use your own target sequences,
  - You must install BLAST,
  - You need to download the reference database.

Running locally

We're interested in codable solutions, so that we can run things in an automated fashion. What do we need to do this?

- Have BLAST+ installed,
- Download the reference genome to use as a target (unless you have your own target genome, of course),
- Have your query sequence,
- Have rBLAST installed in your R installation.

Once we have these pieces assembled we're ready to run BLAST.

Getting BLAST+

You need to have BLAST+ installed even if you are going to work with BLAST through rBLAST.

On a cluster: If you're on a supercomputer like SciNet's Teach cluster or Niagara, there's likely a module for it, which you need to load on the shell command line, before launching R:

```
ejspence@nia-login02 ~>
ejspence@nia-login02 ~> module load gcc
ejspence@nia-login02 ~> module load gmp lmdb
ejspence@nia-login02 ~> module load boost blast+
ejspence@nia-login02 ~>
```

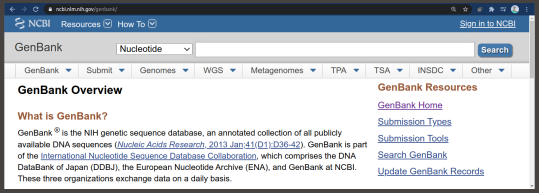
On your own computer: For local installations, go to ftp.ncbi.nlm.nih.gov

Select the version appropriate for your OS and install it, or your OS's package manager may have it (e.g. `sudo apt install ncbi-blast+` on Ubuntu).

Getting a reference genome

You can align any sequence to any other, but mostly people align against reference genomes.

We've already seen that Genbank has such data, and that we can use the 'rentrez' package to download sequences.



To write a script to get reference sequences:

- The data is large (hundreds of GB), and is better downloaded in compressed form, which requires additional, possibly external tools to uncompress. You only want to do this once.
- The data needs to be pre-processed to create the index. This is done by one of the BLAST utilities. You only want to do this once.

Aligning against reference genomes, an aside

As a general rule, if you're using large reference genomes, you don't want to be doing alignment on your laptop.

- These genomes, and associated indices, can take a large amount of storage space.
- Downloading them, and creating the associated indices, takes a fair amount of computation time.
- You will be better served doing your alignment problems on the supercomputer.
- About one third of the computation cycles on the supercomputers are used for alignment and sequencing problems.

We'll do a small example using rBLAST, that is fit for your laptop. But the setup that I will present will be what is appropriate for the Niagara supercomputer.

Installing rBLAST

We now need to install rBLAST.

This really is just a wrapper around BLAST. As such, you must have BLAST+ installed.

As you can, rBLAST is not hosted on CRAN, but rather is stored at r-universe.dev.

Since we're at it, let's load the Biostrings and rentrez libraries as well, as we'll need them later.

```
>
> install.packages('rBLAST',
+       repos = 'https://mhahsler.r-universe.dev')
>
> library(rBLAST)
>
> library(Biostrings)
> library(rentrez)
>
```

Getting a reference genome, continued

Let's do an example. Let's download the reference genome for the pepper plant, *Capsicum annuum*. This isn't too big (960MB), but you'll still need a good internet connection.

For BLAST to work with the database, we need to create an index to the reference genome.

```
> download.file(url.below, 'pepper.fna.gz')
> system('gunzip pepper.fna.gz')
>
> reference <- 'pepper.fna'
>
> makeblastdb(reference, dbtype = 'nucl')
Building a new DB, current time:  03/10/2023 09:50:39
New DB name:  /Users/ejspence/pepper/pepper.fna
New DB title:  pepper.fna
Sequence type:  Nucleotide
Keep MBits:  T
Maximum file size:  3000000000B
Adding sequences from FASTA; added 81202 sequences in
51.3994 seconds.
```

https://ftp.ncbi.nlm.nih.gov/genomes/refseq/plant/Capsicum_annuum/representative/GCF_002878395.1_UCD10Xv1.1/GCF_002878395.1_UCD10Xv1.1_genomic.fna.gz

Getting a reference genome, continued more

Alternatively, you can sometimes download and use pre-processed data that comes with its index prebuilt. The 'nt' genome (which stands for 'Non-redundant nucleoTide') is one such data set. You'll likely use this genome for DNA alignment. For protein alignment, use the 'nr' database.

```
>
> library(stringr)
>
> for (i in 0:87) {
+     filename <- paste('nt.', str_pad(i, 2, pad = '0'), '.tar.gz', sep = '')
+     download.file(paste('ftp://ftp.ncbi.nlm.nih.gov/blast/db/', filename, sep = ''))
+     system(paste('tar -zxf', filename))
+ }
>
```

The *Capsicum annuum* genome does not come pre-indexed.

Commands of BLAST+

BLAST+ has a number of commands. rBLAST will call these for you, but it's good to know what they are and what they do.

Program	Query type	Database/reference type
blastn	nucleotides	nucleotides
blastp	protein	protein
blastx	nucleotides	protein
tblastn	protein	nucleotides
tblastx	nucleotide	nucleotides
psiblast	protein	protein
deltablast	protein	protein

Downloading the target genome

We're going to look for the alignment of the Pun1 gene, as gathered from a Thai hot pepper. The Pun1 gene is one of the genes responsible for the production of capsaicinoids, the alkaloids that cause hot peppers to be hot.

I found this gene by doing a search on the NCBI web site.

```
> clean.fasta <- function(full.fasta.string) {
+     temp <- strsplit(full.fasta.string, '\n')[[1]]
+     temp2 <- temp[-1]
+     return(paste(temp2, collapse = ''))
+ }
>
> thai.pun1 <- entrez_search(db = 'nucleotide',
+     term = 'Thai hot pun1')
> str(thai.pun1)
List of 5 $ids :  chr "55824326"
$count :  int 1
$retmax :  int 1
$QueryTranslation:  chr "Thai[All Fields] AND hot[All
Fields] AND pun1[All Fields]"
$file :Classes 'XMLInternalDocument', 'XMLAbstractDocu-
ment' <externalptr>
- attr(*, "class")= chr [1:2] "esearch" "list"
```

Downloading the target genome, continued

We use rentrez to download the relevant gene sequence.

Taking a look at the FASTA string, we can see that we've downloaded a single gene, with a width of 3752.

```
>
> cap <- entrez_fetch(db = 'nucleotide',
+         id = thai.pun1$ids[1], rettype = 'fasta')
>
> str(cap)
chr ">AY819029.1 Capsicum annuum cultivar
Thai Hot acyltransferase (Pun1) gene, complete
cds\nTCATTAGAAGGTCATACCGCTC"| __truncated__
>
> cap <- DNAStringSet(clean.fasta(cap))
>
> cap
DNAStringSet object of length 1:
   width seq
[1] 3752 TCATTAGAAGGTCATACC...AAGTTTACGGTCAACAA
>
```

Running rBLAST

We now specify the path to the indexed pepper genome. Obviously, don't use the path specified to the right.

Recall that 'blastn' indicates a nucleotide-nucleotide alignment search.

Once the handle to the reference genome has been initialized we can perform the alignment itself.

```
>
> reference <- '/path/to/your/genome/pepper.fna'
>
> bl <- blast(db = reference, type = 'blastn')
>
> cl <- predict(bl, cap)
>
```

Running rBLAST, continued

```
> 
> cl[1:5,]
```

	QueryID	SubjectID	Perc.Ident	Alignment.Length	Mismatches	Gap.Openings
1	Query_1	NC_061112.1	99.742	1164	2	1
2	Query_1	NC_061112.1	89.666	1229	103	13
3	Query_1	NC_061112.1	89.176	1238	102	19
4	Query_1	NC_061112.1	88.502	1235	110	13
5	Query_1	NC_061112.1	89.991	1149	91	14

	Q.start	Q.end	S.start	S.end	E	Bits
1	2588	3751	135885731	135884569	0	2132
2	7	1217	126233055	126231833	0	1544
3	7	1219	148050737	148049507	0	1515
4	9	1219	137073510	137074736	0	1465
5	1	1138	115073321	115074456	0	1463

```
> 
```

Interpreting the rBLAST results

The first entry looks like an interesting alignment. Let's look at that more carefully.

Let's gather the two sequences which match up. First we need to find out where the match came from.

We can use the 'grep' command to find the sequence in the pepper genome that contains the matching sequence.

```
>
> cl$SubjectId[1]
[1] "NC_061112.1"
>
> pepper <- readDNAStringSet('pepper.fna')
>
> pepper.names <- names(pepper)
>
> grep(cl$SubjectId[1], pepper.names)
[1] 2
>
> pepper.names[2]
[1] "NC_061112.1 Capsicum annuum cultivar UCD-10X-F1
chromosome 2, UCD10Xv1.1, whole genome shotgun se-
quence"
>
```

Interpreting the rBLAST results, continued

We now notice something strange. The start of the reference sequence is bigger than the end.

By default BLAST will query both DNA strands. If the start is greater than the stop it indicates that we're looking at the reverse complement of what we're actually aligning.

```
>
> cl$S.start[1]
[1] 135885731
>
> cl$S.end[1]
[1] 135884569
>
> cl$S.end[1] > cl$S.start[1]
[1] FALSE
>
> aligned.pep <- subseq(pepper[[2]], cl$S.end[1],
+                       cl$S.start[1])
>
> aligned.pep <- reverseComplement(aligned.pep)
>
```

Interpreting the rBLAST results, continued more

Now let's get the sub-sequence of our query that was aligned.

In this case the start index is smaller than the end index.

Using summary we can see the 2 mismatched nucleotides for this large subsequence.

```
> aligned.cap <- subseq(cap, cl$Q.start[1],
+                       cl$Q.end[1])
>
> align <- pairwiseAlignment(aligned.cap, aligned.pep)
>
> summary(align)
Scores:
  Min.   1st Qu.   Median   Mean   3rd Qu.   Max.
  2275      2275     2275   2275      2275   2275

Number of matches:
  Min.   1st Qu.   Median   Mean   3rd Qu.   Max.
  1161      1161     1161   1161      1161   1161

Top 2 Mismatch Counts:
     SubjectPosition   Subject   Pattern   Count   Probability
  1              965         C         G       1             1
  2             1107         A         C       1             1
```

Summary

Some notes about alignment:

- Alignment is the comparing of two genetic sequences to one another. Usually a reference genome and a target.
- Various scores are used to determine how good the alignment is. We're usually interested in the 'e-score'; the lower the better.
- For alignment, use existing, fast tools like BLAST.
- rBLAST can interface with BLAST, making it easier to build a pipeline, but you'll need to install BLAST outside of R.
- Reference data can be big: download once and reuse.